ektron

Ektron Developer Reference

Ektron Developer Reference

Ektron 9.40 SP1

Doc. Rev. 1.1

Episerver

Episerver

542 Amherst St.

Nashua, NH 03063

http://www.episerver.com

For the latest version of this manual, go to
http://documentation.ektron.com/cms400/edr/web/edr.htm

Trademarks

Ektron, the Ektron logo and Ektron, are trademarks of Episerver.

Other company and product names may be trademarks of their respective owners.

Table of contents

Ektron Developer Reference

- Framework API (API). Lets you interact with Ektron CMS by providing a highly consistent and discoverable application programming interface (API) for content management operations on Ektron objects.
- Framework UI (UI). Lets you interact with Ektron CMS user controls.
- Templated server controls (TSC). Let you interact with Ektron CMS user controls in a more specialized way than their predecessors, which typically used 1 control to retrieve and display results; templated server controls use 3 controls (InputView, Controller, and ResultsView) to provide functionality.
- Server controls (SC). Drag and drop a server control onto an .aspx page to provide out-of-the-box markup and functionality.
- CMS strategies (Strat). Lets you modify the behavior of Ektron.
- User and content constants (Cnst). Determine constant values when working with Ektron users and content.
- Ektron Markup Language (EkML). Lets you manage the output presentation of server controls.
- *IQueryable data sources for Ektron data types* on page 2539. Provides an IQueryable interface to interact with most core Ektron data types.

Which version can I use?

The Ektron Developer Reference provides information for Ektron versions 8.50 through the current version. The information is labeled as follows throughout the document to identify the versions in which the item is valid. Content that is not tagged is assumed to be valid for 8.50 and higher.

8.50 and higher. Valid for Ektron versions **8.50** and higher.

8.60 and higher. Valid for Ektron versions **8.60** and higher.

8.61 and higher. Valid for Ektron versions **8.6.1** and higher.

8.70 and higher. Valid for Ektron versions **8.70** and higher.

9.00 and higher. Valid for Ektron versions **9.00** and higher.

9.10 and higher. Valid for Ektron versions **9.10** and higher.

9.20 and higher. Valid for Ektron versions **9.20** and higher.

8.60 Additions to the Framework API

- Calendar > WebEventManager > GetEventOccurrenceList (event taxonomy)
- Calendar > WebEventManager > GetEventOccurrenceList (Web event)
- Calendar > WebEventManager > GetList (taxonomy)
- Community > CommunityGroupManager > GetList (UserToCommunityGroupCriteria)
- Community > FriendsManager > Invite (user)
- Community > FriendsManager > IsFriend
- Community > MessageBoardManager > IsUserMessageBoardAdmin (Int64,MessageBoardObjectType,Int64)
- Community > RatingManager > Add
- Community > RatingManager > Delete
- Community > RatingManager > GetItem
- Community > RatingManager > GetList
- Community > RatingManager > GetUserTaxonomy
- Community > RatingManager > Purge
- Community > RatingManager > Purge (by state)
- Community > RatingManager > Update
- Content > ContentManager > Cancel
- Content > ContentManager > CopyContent
- Content > ContentManager > GetContentByHistoryId(Int64)
- Content > ContentManager > GetHistoryList(Int64)
- Content > ContentManager > GetHtmlDifference(String,String)
- Content > ContentManager > MoveContent
- Content > ContentManager > Save(ContentData)
- Content > LibraryManager > GetLibraryItemByContentId(Int64,Int32)
- Flag > FlagDefinitionManager > Add
- Flag > FlagDefinitionManager > Delete
- Flag > FlagDefinitionManager > GetItem
- Flag > FlagDefinitionManager > GetList
- Flag > FlagDefinitionManager > Update
- Organization > FolderManager > AssignMetadata
- Organization > FolderManager > CopyFolder(Int64,Int64,Boolean,Boolean)
- Organization > FolderManager > DisableTaxonomy
- Organization > FolderManager > EnableTaxonomy
- Organization > FolderManager > GetAssignedFolders(Int64)
- Organization > FolderManager > GetAssignedMetadata(Int64)
- Organization > FolderManager > GetSitemapPath(Int64)
- Organization > FolderManager > GetTree(Int64)

- Organization > FolderManager > IsCommunityFolder(Int64)
- Organization > FolderManager > MoveFolder(Int64,Int64,Boolean)
- Organization > FolderManager > Purge(Int64,DateTime,Boolean,Boolean)
- Organization > FolderManager > UnlinkTaxonomy(Int64,Int64)
- Organization > MenuManager > GetMenuList(MenuCriteria)
- Organization > TaxonomyItemManager > Reorder (Int64,Int64,Int32,TaxonomyReorderItemType,Boolean)
- Organization > TaxonomyManager > GetList (TaxonomyCustomPropertyCriteria)
- Organization > TaxonomyManager > ImportTaxonomy(String,String)
- Organization > TaxonomyManager > MoveTaxonomy(Int64,Int64)
- Organization > TaxonomyManager > Reorder(Int64,Int32,Boolean)
- Settings.DxH > DXHUserConnectionManager > Add
- Settings.DxH > DXHUserConnectionManager > Delete
- Settings.DxH > DXHUserConnectionManager > GetItem
- Settings.DxH > DXHUserConnectionManager > GetList
- Settings.DxH > DXHUserConnectionManager > Update
- Settings > PermissionManager > GetUserPermissionForContent (Int64,Int64,Int32)
- Settings > PermissionManager > GetUserPermissionForFolder (Int64,Int64,Int32)
- Settings > PermissionManager > IsLoggedInUserAdmin()
- Settings > SmartformConfigurationManager > GetXmlSearchFieldData(Int64)
- Settings > TaskCommentManager > Add
- Settings > TaskCommentManager > Delete
- Settings > TaskCommentManager > GetItem
- Settings > TaskCommentManager > GetList
- Settings > TaskCommentManager > Update
- User > UserGroupManager > GetUserList(Int64)
- User > UserManager > ActivateUserAccount (account ID)
- User > UserManager > ActivateUserAccount (user)
- User > UserManager > ActivateUserAccounts (user ID)
- User > UserManager > GetList (property)
- User > UserManager > LockUser(Int64)
- User > UserManager > ResetPassword(String,Int64)
- User > UserManager > ResetPassword(String,String,String)
- User > UserManager > UnlockUser(Int64)

8.60 Additions to the templated server controls

- Content
- MenuView
- AccessPoint

8.6.1 Additions to the Framework API

- Settings.UrlAliasing > AliasManager > Add
- Settings.UrlAliasing > AliasManager > Delete
- Settings.UrlAliasing > AliasManager > DeleteAll(). Deletes all aliases.
- Settings.UrlAliasing > AliasManager > GetItem
- Settings.UrlAliasing > AliasManager > GetList
- Settings.UrlAliasing > AliasManager > GetTarget
- Settings.UrlAliasing > AliasManager > IsValidEktronAlias
- Settings.UrlAliasing > AliasManager > Update
- Settings.UrlAliasing > AliasRuleManager > Add
- Settings.UrlAliasing > AliasRuleManager > Delete
- Settings.UrlAliasing > AliasRuleManager > DeleteAliases
- Settings.UrlAliasing > AliasRuleManager > DisableAliases
- Settings.UrlAliasing > AliasRuleManager > DisableAll
- Settings.UrlAliasing > AliasRuleManager > EnableAliases
- Settings.UrlAliasing > AliasRuleManager > GetItem
- Settings.UrlAliasing > AliasRuleManager > GetList (AliasRule)
- Settings.UrlAliasing > AliasRuleManager > GetList (objects)
- Settings.UrlAliasing > AliasRuleManager > Update
- Settings.UrlAliasing > AliasSettingManager > AddFileExtension
- Settings.UrlAliasing > AliasSettingManager > ClearCache
- Settings.UrlAliasing > AliasSettingManager > DeleteExtension
- Settings.UrlAliasing > AliasSettingManager > Get ()
- Settings.UrlAliasing > AliasSettingManager > Get (name and site)
- Settings.UrlAliasing > AliasSettingManager > Get (site)
- Settings.UrlAliasing > AliasSettingManager > GetAllExtensions()
- Settings.UrlAliasing > AliasSettingManager > GetAllExtensions (ID)
- Settings.UrlAliasing > AliasSettingManager > GetAllExtensionsOnly()
- Settings.UrlAliasing > AliasSettingManager > GetAllExtensionsOnly (site ID)
- Settings.UrlAliasing > AliasSettingManager > GetExtension (ID)
- Settings.UrlAliasing > AliasSettingManager > Update (aliasSetting)
- Settings.UrlAliasing > AliasSettingManager > Update (name and ID)
- Settings.UrlAliasing > RedirectManager > Add
- Settings.UrlAliasing > RedirectManager > Delete
- Settings.UrlAliasing > RedirectManager > DeleteAll
- Settings.UrlAliasing > RedirectManager > GetItem

- Settings.UrlAliasing > RedirectManager > GetList
- Settings.UrlAliasing > RedirectManager > GetTarget
- Settings.UrlAliasing > RedirectManager > Update

8.6.1 Additions to the Framework UI

- Captcha

8.70 Additions to the Framework API

- Content > ContentManager > GetAssignedTaxonomyList(Int64,Int32)

9.00 Additions to the Framework API

- Commerce > AddressManager > Add (address)
- Commerce > AddressManager > Add (customer ID)
- Commerce > AddressManager > Delete
- Commerce > AddressManager > GetItem
- Commerce > AddressManager > GetList
- Commerce > AddressManager > Update
- Commerce > AddressManager > UpdateOrderAddress
- Commerce > BasketItemManager > Add
- Commerce > BasketItemManager > Delete
- Commerce > BasketItemManager > GetKitConfiguration (shopper)
- Commerce > BasketItemManager > GetKitConfiguration(user)
- Commerce > BasketItemManager > Update
- Commerce > BasketItemManager > UpdateQuantity
- Commerce > BasketManager > Add
- Commerce > BasketManager > ApplyCoupon
- Commerce > BasketManager > Delete
- Commerce > BasketManager > Empty
- Commerce > BasketManager > GetAppliedCoupons
- Commerce > BasketManager > GetDefault (default)
- Commerce > BasketManager > GetDefault (shopper)
- Commerce > BasketManager > GetDefault (user)
- Commerce > BasketManager > GetItem
- Commerce > BasketManager > GetList
- Commerce > BasketManager > RemoveCoupon
- Commerce > BasketManager > SetDefault
- Commerce > BasketManager > Update
- Commerce > CatalogEntryManager > Add
- Commerce > CatalogEntryManager > Cancel
- Commerce > CatalogEntryManager > CheckIn

- Commerce > CatalogEntryManager > CheckOut
- Commerce > CatalogEntryManager > Delete
- Commerce > CatalogEntryManager > DisableInventory
- Commerce > CatalogEntryManager > EnableInventory
- Commerce > CatalogEntryManager > GetItem
- Commerce > CatalogEntryManager > GetList (catalog entries)
- Commerce > CatalogEntryManager > GetList (entry attribute)
- Commerce > CatalogEntryManager > Restore
- Commerce > CatalogEntryManager > Submit
- Commerce > CatalogEntryManager > Update
- Commerce > CountryManager > Add
- Commerce > CountryManager > CanDelete
- Commerce > CountryManager > Delete
- Commerce > CountryManager > GetItem
- Commerce > CountryManager > GetList
- Commerce > CountryManager > Update
- Commerce > CouponManager > Add
- Commerce > CouponManager > AddCouponToObject
- Commerce > CouponManager > Deactivate
- Commerce > CouponManager > Delete
- Commerce > CouponManager > DeleteCouponApplications
- Commerce > CouponManager > GetCatalogList
- Commerce > CouponManager > GetCouponId
- Commerce > CouponManager > GetItem
- Commerce > CouponManager > GetList
- Commerce > CouponManager > GetProductList
- Commerce > CouponManager > GetTaxonomyList
- Commerce > CouponManager > IsCouponApplicabletoSubscriptions
- Commerce > CouponManager > IsCouponAppliedToObject
- Commerce > CouponManager > IsCouponUsedForBasket
- Commerce > CouponManager > IsCouponUsedForOrder
- Commerce > CouponManager > SaveCouponApplications
- Commerce > CouponManager > Update
- Commerce > CouponManager > Validate (coupon)
- Commerce > CouponManager > Validate (product)
- Commerce > CreditCardTypeManager > Add
- Commerce > CreditCardTypeManager > Delete
- Commerce > CreditCardTypeManager > GetAcceptedCreditCardList
- Commerce > CreditCardTypeManager > GetItem

- Commerce > CreditCardTypeManager > GetList
- Commerce > CreditCardTypeManager > IsCardValid
- Commerce > CreditCardTypeManager > IsDateValid
- Commerce > CreditCardTypeManager > Update
- Commerce > CurrencyManager > Add
- Commerce > CurrencyManager > CanDelete
- Commerce > CurrencyManager > Delete
- Commerce > CurrencyManager > Delete
- Commerce > CurrencyManager > GetActiveCurrencyList
- Commerce > CurrencyManager > GetCurrentCurrency
- Commerce > CurrencyManager > GetDefaultCurrency
- Commerce > CurrencyManager > GetItem
- Commerce > CurrencyManager > GetList
- Commerce > CurrencyManager > Update
- Commerce > CustomerManager > Add (billing address)
- Commerce > CustomerManager > Add (customer ID)
- Commerce > CustomerManager > Add (shipping address)
- Commerce > CustomerManager > Delete
- Commerce > CustomerManager > GetItem
- Commerce > CustomerManager > GetList
- Commerce > CustomerManager > IsUserNameExists
- Commerce > CustomerManager > SetBillingAddress
- Commerce > CustomerManager > SetShippingAddress
- Commerce > CustomerManager > Update
- Commerce > ExchangeRateManager > Add
- Commerce > ExchangeRateManager > Delete
- Commerce > ExchangeRateManager > GetCurrentExchangeRate
- Commerce > ExchangeRateManager > GetCurrentExchangeRate (ID)
- Commerce > ExchangeRateManager > GetCurrentList
- Commerce > ExchangeRateManager > GetList
- Commerce > ExchangeRateManager > Update
- Commerce > InventoryManager > Add
- Commerce > InventoryManager > DecreaseStockLevel
- Commerce > InventoryManager > GetItem
- Commerce > InventoryManager > GetList
- Commerce > InventoryManager > GetReorderLevel
- Commerce > InventoryManager > GetUnitsInStock
- Commerce > InventoryManager > GetUnitsOnOrder
- Commerce > InventoryManager > IncreaseStockLevel

- Commerce > InventoryManager > IsItemAvailable
- Commerce > InventoryManager > Update
- Commerce > OrderManager > ApplyCoupon
- Commerce > OrderManager > Capture (order)
- Commerce > OrderManager > Capture (payment)
- Commerce > OrderManager > Delete
- Commerce > OrderManager > GetItem
- Commerce > OrderManager > GetList
- Commerce > OrderManager > GetOrderPaymentList
- Commerce > OrderManager > GetStatus
- Commerce > OrderManager > PlaceOrder
- Commerce > OrderManager > SetFraud
- Commerce > OrderManager > SetHold
- Commerce > OrderManager > SetStatus
- Commerce > OrderManager > SetTrackingNumber
- Commerce > OrderManager > Update
- Commerce > PackageManager > Add
- Commerce > PackageManager > Delete
- Commerce > PackageManager > GetItem
- Commerce > PackageManager > GetList
- Commerce > PackageManager > Update
- Commerce > PasswordHistoryManager > Add
- Commerce > PasswordHistoryManager > Delete
- Commerce > PasswordHistoryManager > GetItem
- Commerce > PasswordHistoryManager > GetList
- Commerce > PasswordHistoryManager > GetRecentPasswords
- Commerce > PasswordHistoryManager > MatchesRecentPassword
- Commerce > PasswordHistoryManager > Purge
- Commerce > PasswordHistoryManager > Update
- Commerce > PaymentGatewayManager > Add
- Commerce > PaymentGatewayManager > Delete
- Commerce > PaymentGatewayManager > GetDefault
- Commerce > PaymentGatewayManager > GetItem
- Commerce > PaymentGatewayManager > GetList
- Commerce > PaymentGatewayManager > SetDefault
- Commerce > PaymentGatewayManager > Update
- Commerce > ProductTypeManager > Add
- Commerce > ProductTypeManager > Delete
- Commerce > ProductTypeManager > GetItem

- Commerce > ProductTypeManager > GetList (folder ID)
- Commerce > ProductTypeManager > GetList (product type criteria)
- Commerce > ProductTypeManager > Update
- Commerce > RecommendationManager > Delete
- Commerce > RecommendationManager > DeleteByEntry
- Commerce > RecommendationManager > GetItem
- Commerce > RecommendationManager > GetList
- Commerce > RecommendationManager > UpdateCrossSell
- Commerce > RecommendationManager > UpdateUpSell
- Commerce > RegionManager > Add
- Commerce > RegionManager > CanDelete
- Commerce > RegionManager > Delete
- Commerce > RegionManager > GetItem
- Commerce > RegionManager > GetList
- Commerce > RegionManager > Update
- Commerce > ShippingMethodManager > Add
- Commerce > ShippingMethodManager > Delete
- Commerce > ShippingMethodManager > GetItem
- Commerce > ShippingMethodManager > GetList
- Commerce > ShippingMethodManager > Update
- Commerce > TaxClassManager > Add
- Commerce > TaxClassManager > Delete
- Commerce > TaxClassManager > GetItem
- Commerce > TaxClassManager > GetList
- Commerce > TaxClassManager > IsUsed
- Commerce > TaxClassManager > Update
- Commerce > TaxRateManager > Add
- Commerce > TaxRateManager > Delete
- Commerce > TaxRateManager > DeleteByCountry
- Commerce > TaxRateManager > DeleteByRegion
- Commerce > TaxRateManager > GetApplicableTaxRate
- Commerce > TaxRateManager > GetApplicableTaxRateList
- Commerce > TaxRateManager > GetItem
- Commerce > TaxRateManager > GetList
- Commerce > TaxRateManager > Update
- Commerce > TaxTypeManager > Add
- Commerce > TaxTypeManager > Delete
- Commerce > TaxTypeManager > GetItem
- Commerce > TaxTypeManager > GetList

- Commerce > TaxTypeManager > Update
- Commerce > WarehouseManager > Add
- Commerce > WarehouseManager > Delete
- Commerce > WarehouseManager > GetDefault
- Commerce > WarehouseManager > GetItem
- Commerce > WarehouseManager > GetList
- Commerce > WarehouseManager > SetDefault
- Commerce > WarehouseManager > Update
- Content > ContentManager > Approve(Int64)
- Organization > FolderManager > UnAssignTemplate(Int64,Int64)
- Organization > MenuManager > GetMenuItemList(MenuItemCriteria)
- Search > QueryPropositionManager > GetQueryCompletions
- Search > QueryPropositionManager > GetQueryCompletions (QueryCompletionRequest)
- Search > QueryPropositionManager > GetQueryCompletions (String,Int32,QueryCompletionSortOrder)
- Search > QueryPropositionManager > GetQuerySuggestions
- Search > QueryPropositionManager > GetQuerySuggestions (QuerySuggestionRequest)
- User > UserManager > GetItem(Int64,Boolean)

9.00 Additions to the Framework UI

- InfoTip
- Tree
- Wizard

9.10 Additions to the Framework UI

- Tree > ContextMenu Tree
- Tree > FolderTree with Ajax
- Tree > Custom-Processor

1

Framework API

The Framework Application Programming Interface (API) lets you interact with Ektron CMS. Learn more about the Framework API from the following video:The Ektron Framework API and Microsoft LINQ

The Framework API promotes *discoverability* and *consistency*. Discoverability means that a developer should be able to guess the namespace, object name, and method to use without having to read through a lot of documentation. A well-designed API lets you use contextual clues by the naming of objects and Intellisense to make your way through a situation. Consistency makes the API function in similar ways for similar functions.

The API namespace consists of the following parts:

```
Company.System.Framework.Manager.Class.Method(Object)
```

For example:

```
Ektron.Cms.Framework.Activity.ActivityCommentManager.Add
(Ektron.Cms.Activity.ActivityCommentData)
```

The Framework API interacts with the underlying permission model. When you create a framework object to perform actions, the constructor has an optional parameter to specify the access mode for that object. For example, when you create a new Content object, the code snippet looks as follows:

```
Ektron.Cms.Framework.Core.Content.Content content
  = new Ektron.Cms.Framework.Core.Content.Content(
     Ektron.CMS.Framework.ApiAccessMode.Admin);
```

The snippet contains the optional parameter specifying the access mode, which in this case is set to `Admin`. Only administrators and currently logged-in users are the 2 valid options in the enumeration.

- `Ektron.CMS.Framework.ApiAccessMode.Admin`. The Admin mode specifies permissions should be ignored for all actions undertaken by this object.
- `Ektron.CMS.Framework.ApiAccessMode.LoggedInUser`. The object queries and works within the permission set for the current user.

By switching between these 2 options, you can circumvent the permissions model when you have a task that requires it.

If you skip this parameter, the default behavior is to act as the currently logged-in user. When objects are created as the currently logged-in user, the system properly maintains user attributions in the history for that item.

CRUD operations

Code snippets demonstrate practical used for performing create, read, update, and delete (CRUD) operations on CMS objects, such as how to create content, retrieve a list of users, and delete folders and so on. You can use the code on your own website and then customize it.

The following sections outline how to use CRUD operations with content, but the methodology is the same for all objects that are exposed by the Framework API.

Create

The `Add()` method creates a content object. By taking in a data object and storing the new data in the database. In the Content object, the data object has type `Ektron.Cms.ContentData` and is passed by reference, which means fields are updated in place after the content is added to the system. The following code snippet adds a piece of content to the root folder with the language set to English.

```
//Create the Content object set to observer permissions
Ektron.Cms.Framework.Core.Content.Content ContentAPI
  = new Ektron.Cms.Framework.Core.Content.Content();

//Set up the ContentData object
Ektron.Cms.ContentData newContent = New Ektron.Cms.ContentData();
newContent.LanguageId = 1033;
newContent.FolderId = 0;
newContent.Title = "Content added through through the Framework API";
newContent.Teaser = "The summary for my content";
newContent.Html = "<p>The HTML for programatically added content</p>";

//Add the content
ContentAPI.Add(newContent)

//Output the new content ID
Response.Writer(newcontent.Id.ToString());
```

Retrieve

GetItem

The `GetItem()` method retrieves and returns a single data object. The following code snippet retrieves the content item with ID 30 in the current language.

NOTE: This code respects the permission settings of the currently logged-in user, so if you lack permission to access the content, the object returned will be empty unless you instansiate the object in `ApiAccessMode.Admin`.

```
//Create the Content Object set to observe permissions
Ektron.Cms.Framework.Core.Content.Content ContentAPI
  = new Ektron.Cms.Framework.Core.Content.Content();

//Retrieve the content
Ektron.Cms.ContentData contentData;
contentData = ContentAPI.GetItem(30);

//Output the retrieved item's content
Response.Write(contentData.Html);
```

GetList

The `GetList()` method retrieves and returns a list of data objects using `criteria` objects. Criteria objects accept items from enumeration containing fields against

which to search, along with an operator and an operand. Use `AddFilter()` to add these tuples (a set of objects that belong together).

The following code snippet retrieves the content item with ID 30 in the current language.

```
//Create the Content Object set to observe permissions
Ektron.Cms.Framework.Core.Content.Content ContentAPI
  = new Ektron.Cms.Framework.Core.Content.Content();

//Create the criteria object
Ektron.Cms.Common.Criteria<Ektron.Cms.Common.ContentProperty> myCriteria
  = new Ektron.Cms.Common.Criteria<Ektron.Cms.Common.ContentProperty>();

//Add a filter to specify the root folder
myCriteria.AddFilter(
  Ektron.Cms.Common.ContentProperty.FolderId,
  Ektron.Cms.Common.CriteriaFilterOperator.EqualTo,
  0);

//Add a filter to retrieve only published items
myCriteria.AddFilter(
  Ektron.Cms.Common.ContentProperty.IsPublished,
  Ektron.Cms.Common.CriteriaFilterOperator.EqualTo,
  true);

//Add a filter to retrieve only items that contain the word "TitleSearch"
//in the title
myCriteria.AddFilter(
  Ektron.Cms.Common.ContentProperty.Title,
  Ektron.Cms.Common.CriteriaFilterOperator.Contains,
  "TitleSearch");

//Create the output object
List<Ektron.Cms.ContentData> resultList;

//Retrieve the results
resultList = ContentAPI.GetList(myCriteria);
```

Criteria use for GetList methods

This section applies to all GetList methods.

The DataProperty enum works with its criteria class and defines properties that the Manager class uses to filter and sort. Specify zero (0) or more criteria to retrieve the objects you want. The criteria object also lets you set the number of items to return, define paging data, and modify sorting.

Criteria objects

- `Condition`. The Logical operator to apply to the filters. The default value is "AND". The criteria condition controls the operator for the default filter group and the AND-ing\Or-ing the filter groups together.

  ```
  public Ektron.Cms.Common.LogicalOperation Condition { set; get; }
  ```

- `FilterGroups`. Gets a list of filter groups. Filter groups can be created to do filter grouping (that is, parentheses in the where clause). The groups use the criteria's Condition like other filters, but can use their own condition internally.

```
public
System.Collections.Generic.List<CriteriaFilterGroup<ActivityCommentProperty>>
FilterGroups { get; }
```

- `Filters`. Lists of filters defined in this criteria object. To add new Filters, you can add them here or use the AddFilter method (recommended).

```
public System.Collections.Generic.List<CriteriaFilter<ActivityCommentProperty>>
Filters { get; }
```

- `GroupByField`. The group by field to apply to the results.

```
public Ektron.Cms.Activity.ActivityCommentProperty GroupByField { set; get; }
```

- `OrderByDirection`. Sets the order direction (ascending or descending).

```
public EkEnumeration.OrderByDirection OrderByDirection { set; get; }
```

- `OrderByField` . Set the property used to order the results.

```
public Ektron.Cms.Activity.ActivityCommentProperty OrderByField { set; get; }
```

- `PagingInfo`. By default, page size is 50 for improved performance. Example paging Info:

```
PagingInfo pageinfo = new PagingInfo();
pageinfo.CurrentPage=1;
pageinfo.RecordsPerPage = 100;
criteria.PagingInfo = pageinfo;
```

Constructor(property, orderbyDirection)

Sorts results by a property, in ascending or descending order.

AddFilter(property, operator, value)

Filters results based on conditions defined in this method.

```
criteria.AddFilter(Property.ObjectType, CriteriaFilterOperator.EqualTo, Objectvalue);
```

LogicalOperation condition

Filters the array of items based on the logical condition set in the AddFilter properties.

```
criteria.AddFilter(Property.ObjectType, CriteriaFilterOperator.EqualTo,Objectvalue);
```

List<CriteriaFilterGroup> FilterGroups

Binds data to a listview or gridview.

Update

The `Update()` method updates a content object, by taking in a data object and storing the modified data in the database. You first retrieve the existing item and the modify the properties. The following code snippet retrieves the content with ID 30 and updates the `Title` property before saving it back to the database.

```
//Create the Content Object set to observe permissions
Ektron.Cms.Framework.Core.Content.Content ContentAPI
  = new Ektron.Cms.Framework.Core.Content.Content();

//Retrieve the content
Ektron.Cms.ContentData myContent;
myContent = ContentAPI.GetItem(30);

//Update a field in the content item
myContent.Title - "This was updated";

//Save the update
ContentAPI.Update(myContent);
```

Delete

The `Delete()` method removes an item from the database. This code respects the permission settings of the currently logged-in user, so if you need permission to delete the content, you must instansiate the object in `ApiAccessMode.Admin`. The following code snippet deletes the item with ID 30.

```
//Create the Content Object set to observe permissions
Ektron.Cms.Framework.Core.Content.Content ContentAPI
  = new Ektron.Cms.Framework.Core.Content.Content();

//Delete the content item
ContentAPI.Delete(30);
```

API classes and managers

The Framework API has the following namespace categories (bold) of managers (links).

Activity. Manages user feedback activity.

- *ActivityCommentManager* on page 163. Manages user comments that appear in the activity stream.
- *ActivityManager* on page 176. Manages user activity items that appear in the activity stream.
- *ActivityStreamManager* on page 211. Manages gets post from and sends posts to the activity stream.
- *ActivityTypeManager* on page 225. Manages activity types that appear in the activity stream.

Calendar. Manages schedules and events.

- *WebCalendarManager* on page 239. Manages calendars.
- *WebEventManager* on page 259. Manages schedules that appear on user calendars.

Commerce. Manages commerce exchange on the Web.

IMPORTANT: The Commerce category is valid for versions 9.00 and up.

- *AddressManager* on page 324. Manages addresses in the CMS.
- *BasketItemManager* on page 345. Manages product items in the basket.
- *BasketManager* on page 360. Manipulates the baskets in e-commerce.
- *CatalogEntryManager* on page 387. Manipulates the catalog entry data in CMS.
- *CountryManager* on page 419. Manipulates the country data in the CMS.
- *CouponManager* on page 435. Manipultes the coupon data in the CMS. It can be used for add, edit, and get the coupon data object.
- *CreditCardTypeManager* on page 480. Manages credit card types in the CMS.
- *CurrencyManager* on page 499. Manages commerce currencies in the CMS.
- *CustomerManager* on page 519. Manipulates e-commerce customer details in CMS.
- *ExchangeRateManager* on page 550. Manages commerce exchange rates in the CMS.
- *InventoryManager* on page 570. Manages inventory in the CMS.
- *OrderManager* on page 591. Manages commerce orders in the CMS.
- *PackageManager* on page 628. Manages commerce packages in the CMS.
- *PasswordHistoryManager* on page 644. Manages password histories.
- *PaymentGatewayManager* on page 665. Manages the commerce paymentgateway in the CMS.
- *ProductTypeManager* on page 683. Manipulates the catalog's product type data in CMS.
- *RecommendationManager* on page 700. Manipulates the recommendated items to the catalog entry.
- *RegionManager* on page 715. Manages regions in the CMS.
- *ShippingMethodManager* on page 730. Manages shipping methods in CMS.
- *TaxClassManager* on page 743. Manages tax class in the CMS.
- *TaxRateManager* on page 755. Manages commerce tax rates in the CMS.
- *TaxTypeManager* on page 780. Manages commerce tax types in the CMS.
- *WarehouseManager* on page 792. Manages commerce warehouses in the CMS.

Community. Manages social media aspects of a website.

- *CommunityGroupManager* on page 812. Manages groups of users.
- *FavoriteManager* on page 867. Manages a list of Ektron CMS content and external Web links (URLs) that a user has designated as favorite content.
- *FavoriteTaxonomyManager* on page 882. Manages user favorite taxonomies.
- *FlagManager* on page 896. Manages flags that are assigned to content to provide feedback.
- *FriendsManager* on page 910. Manages colleague connections (friends).
- *FriendsTaxonomyManager* on page 939. Manages taxonomies of colleague connections (friends).
- *MessageBoardManager* on page 953. Manages items that are posted on a message board.

- *MicromessageManager* on page 979. Manages user status messages that are posted to an activity stream.
- *PrivateMessageManager* on page 1005. Manages messages between users.
- *TagManager* on page 1042. Manages tags.

Content. Manages Web content.

- *AssetManager* on page 1061. Manages assets, which are files that are created outside of CMS.
- *ContentManager* on page 1076. Manages content (including HTML, text, documents, and digital media).
- *ContentRatingManager* on page 1124. Manages user valuations on content.
- *LibraryManager* on page 1142. Manages libraries, which stores images, files, quicklinks, and hyperlinks that can be inserted into editor content.
- *MetadataTypeManager* on page 1157. Manages metadata types, such as title and other artifacts.
- *TemplateManager* on page 1172. Manages templates, which typically includes page headers, footers, and placeholders for content, forms, summaries, calendars, collections, or other page elements.

Flag. Manages flags on content.

- *FlagDefinitionManager* on page 1187

Notifications. Manages information that is sent to users.

- *NotificationManager* on page 1200

Organization. Manages things that help organize content.

- *CollectionManager* on page 1210. Manages a list of content.
- *FolderManager* on page 1231. Manages Ektron CMS folders.
- *MenuManager* on page 1263. Manages drop down menus for your Web site.
- *TaxonomyItemManager* on page 1294. Manages taxonomy items.
- *TaxonomyManager* on page 1310. Manages taxonomies.

Search. Manages online searches.

- *SearchManager* on page 1331. Provides access to query the index of a configured search provider.
- *QueryPropositionManager* on page 1341. Provides access to query the index of a configured search provider.
- *Search Data Classes* on page 1346
- *Search Expressions* on page 1389
- *Search Exceptions* on page 1384
- *Search Abstract Classes and Interfaces* on page 1412

Settings. Manages CMS settings.

- *CmsMessageManager* on page 1424. Manages messages in the Ektron CMS.
- *CmsMessageTypeManager* on page 1442. Manages message types in the Ektron CMS.
- *CustomPropertyManager* on page 1456. Manages user-specified properties to create custom fields.

- *DXH* on page 1469. Sub-category of Settings.
  - *DXHUserConnectionManager* on page 1469. Manages connections between the Digital Experience Hub (DXH) and other applications.
- *Notifications* on page 1479. Sub-category of Settings.
  - *NotificationAgentSettingManager* on page 1480. Manages notification agents.
  - *NotificationPreferenceManager* on page 1498. Manages preferences for notification agents.
  - *NotificationPublishPreferenceManager* on page 1523. Manages preferences for notification agents that can be published.
  - *UserNotificationSettingManager* on page 1535. Manages user-specified notifications.
- *PermissionManager* on page 1552. Manages user permissions.
- *SmartformConfigurationManager* on page 1577. Manages Smart Form configurations.
- *TaskCategoryManager* on page 1600. Manages task categories.
- *TaskCommentManager* on page 1593. Manages task comments.
- *TaskManager* on page 1612. Manages tasks.
- *TaxonomyCustomPropertyManager* on page 1631. Manages user-specified taxonomy properties.
- *UrlAliasing* on page 1649. Sub-category of Settings.
  - *AliasManager* on page 1650. Manages aliases.
  - *AliasRuleManager* on page 1673. Manages alias rules.
  - *AliasSettingManager* on page 1704. Manages alias settings.
  - *AutoAliasManager* on page 1736. Manages automatic aliases, which lets you assign an alias to several content items at once.
  - *CommonAliasManager* on page 1755. Gets alias and target data that is common to any alias.
  - *CommunityAliasManager* on page 1759. Manages aliases that apply to communities.
  - *ManualAliasManager* on page 1776. Manages manual aliases.
  - *RedirectManager* on page 1792. Manages URL redirect addresses.
  - *RegExAliasManager* on page 1812. Manages regular expressions that replace URLs that cannot be read.

User. Manages user information.

- *UserGroupManager* on page 1830. Manages groups.
- *UserManager* on page 1854. Manages users.

Activity

8.50 and higher

The Activity category has the following classes:

- *ActivityCommentManager* on the facing page. Manages user comments that appear in the activity stream.
- *ActivityManager* on page 176. Manages user activity items that appear in the activity stream.
- *ActivityStreamManager* on page 211. Manages gets post from and sends posts to the activity stream.
- *ActivityTypeManager* on page 225. Manages activity types that appear in the activity stream.

ActivityCommentManager

8.50 and higher

The ActivityCommentManager class manages user comments that appear in the activity stream.

Namespace

```
Ektron.Cms.Framework.Activity.ActivityCommentManager
```

Constructors

- `ActivityCommentManager()`
- `ActivityCommentManager(ApiAccessMode)`

Properties

- `ActivityCommentService`. Gets instance to IActivityStream.
- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Activity.ActivityCommentData)
```

Adds activity comment data to the activity stream.

This method takes one argument for the ActivityCommentData data class. When you add the ActivityCommentData, the method returns a new custom ActivityCommentData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity ID
- * Activity Comment

Parameters

- `comment`. Activity comment to add.

Returns

- Returns the custom CmsData object added.

Add .aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxActivityIDLabel" AssociatedControlID="uxActivityID"
CssClass="span-4 last"
      runat="server" Text="*Activity ID :" />
    <ektronUI:TextField ID="uxActivityID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
      Text="0" />
  </li>
  <li class="clearfix">
  <ektronUI:Label ID="uxActivityCommentLabel" AssociatedControlID="uxActivityComment"
    CssClass="span-4 last" runat="server" Text="*Activity Comment :" />
  <ektronUI:TextField ID="uxActivityComment" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
    Text="" />
  </li>
  <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
  <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

Add .aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

Add .aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
{
  try
  {
    ActivityCommentManager activityCommentManager = new ActivityCommentManager();
    //Check whether user logged in or not
    if (activityCommentManager.UserId > 0)
    {
      ActivityManager activityManager = new ActivityManager();
      long activityID = Convert.ToInt64(uxActivityID.Text);
      //Check the ActivityDetails
      ActivityData activityData = activityManager.GetItem(activityID);

      if (activityData != null)
      {
        //Set the Activity Comment Details
        ActivityCommentData activityCommentData = new ActivityCommentData()
        {
          ActivityId = activityID,
          Comment = uxActivityComment.Text,
          DateCreated = DateTime.Now,
          UserId = activityCommentManager.UserId,
        };
        //Add a New Activity Comment with given activityDetails.
        activityCommentManager.Add(activityCommentData);
        MessageUtilities.UpdateMessage(uxMessage, "Activity Comment Saved with Id " +
activityCommentData.Id, Message.DisplayModes.Success);
      }
      else
      {
        MessageUtilities.UpdateMessage(uxMessage,"Please Enter Valid Activity ID and Try
again.", Message.DisplayModes.Error);
      }
    }
    else
    {  MessageUtilities.UpdateMessage(uxMessage, "Please Login and try again.",
Message.DisplayModes.Error); }
    uxPageMultiView.SetActiveView(uxViewMessage);
  }
  catch (Exception ex)
  {
    MessageUtilities.UpdateMessage(uxMessage, ex.Message, Message.DisplayModes.Error);
    uxPageMultiView.SetActiveView(uxViewMessage);
  }
}
```

Delete

```
Delete(System.Int64)
```

Deletes an activity comment from the activity stream.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity Comment ID

Parameters

- `id`. The ID of activity comment to delete.

Remarks

Validate the item using `GetItem()` before deleting.

Delete .aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxActivityCommentIDLabel"
AssociatedControlID="uxActivityCommentID" CssClass="span-4 last"
      runat="server" Text="*ActivityComment ID :" />
    <ektronUI:TextField ID="uxActivityCommentID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
      Text="1" />
  </li>
  <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
  <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

Add .aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

Add .aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
{
  try
  {
    ActivityCommentManager activityCommentManager = new ActivityCommentManager();
    //Check whether user logged in or not
    if (activityCommentManager.UserId > 0)
    {
      long activityCommentID = long.Parse(uxActivityCommentID.Text);
      //Check, Entered  activityCommentID valid or Not
      ActivityCommentData activityCommentData = activityCommentManager.GetItem
(activityCommentID);
      //Check the Activity Comment Data
      if (activityCommentData != null)
      {   //Delete Activity Comment ID from CMS
        activityCommentManager.Delete(activityCommentID);
        MessageUtilities.UpdateMessage(uxMessage, "Activity Comment ID " +
activityCommentID + " has been Deleted.", Message.DisplayModes.Success);
      }
      else { MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
Comment ID.", Message.DisplayModes.Error); }
    }
    else
    { MessageUtilities.UpdateMessage(uxMessage, "Please Login and try again.",
Message.DisplayModes.Error); }
    uxPageMultiView.SetActiveView(uxViewMessage);
  }
  catch (Exception ex)
  {
    MessageUtilities.UpdateMessage(uxMessage, ex.Message, Message.DisplayModes.Error);
    uxPageMultiView.SetActiveView(uxViewMessage);
  }
}
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific ActivityCommentData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The ID of activity comment to retrieve.

GetItem .asxp code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxActivityCommentIDLabel"
AssociatedControlID="uxActivityCommentID"
      CssClass="span-4 last" runat="server" Text="*ActivityComment ID :" />
    <ektronUI:TextField ID="uxActivityCommentID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
      Text="1" />
  </li>
  <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
  <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
  <li class="clearfix">
    <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
  </li>
  <li class="clearfix">
    <asp:Literal ID="uxActivityID" runat="server"></asp:Literal>
  </li>
  <li class="clearfix">
    <asp:Literal ID="uxComment" runat="server"></asp:Literal>
  </li>
  <li class="clearfix">
    <asp:Literal ID="uxUserID" runat="server"></asp:Literal>
  </li>
  <li class="clearfix">
    <asp:Literal ID="uxDateCreated" runat="server"></asp:Literal>
  </li>
</ol>
```

GetItem .aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

GetItem .aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
{
  try
  {
    ActivityCommentManager activityCommentManager = new ActivityCommentManager();
    long activityCommentID = long.Parse(uxActivityCommentID.Text);
```

```
    //Check, Entered  activityCommentID valid or Not
    ActivityCommentData activityCommentData = activityCommentManager.GetItem
(activityCommentID);
    //Check the Activity Comment Data
    if (activityCommentData != null)
    {   //Display the Activity Comment Details.
      MessageUtilities.UpdateMessage(uxMessage,"Activity Comment Details for ID " +
activityCommentID + " are below", Message.DisplayModes.Success);
      uxActivityID.Text = "Activity ID = " + activityCommentData.ActivityId;
      uxComment.Text = "Activity Comment = " + activityCommentData.Comment;
      uxUserID.Text = "User ID = " + activityCommentData.UserId;
      uxDateCreated.Text = "DateCreated = " + activityCommentData.DateCreated;
    }
    else { MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
Comment ID.", Message.DisplayModes.Error); }

    uxPageMultiView.SetActiveView(uxViewMessage);
  }
  catch (Exception ex)
  {
    MessageUtilities.UpdateMessage(uxMessage, ex.Message, Message.DisplayModes.Error);
    uxPageMultiView.SetActiveView(uxViewMessage);
  }
}
```

GetList

```
GetList(Ektron.Cms.Activity.ActivityCommentCriteria)
```

Retrieves lists of objects through the GetList(ActivityCommentCriteria) method.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Activity Property
- * Object Value

Parameters

- `criteria`. Criteria used to filter results.

See *Criteria use for GetList methods* on page 156 for additional information.

GetList.aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxActivityPropertyLabel"
AssociatedControlID="uxActivityProperty"
      CssClass="span-3 last" runat="server" Text="Activity Property :" />
```

```
    <asp:DropDownList ID="uxActivityProperty" runat="server">
      <asp:ListItem>UserId</asp:ListItem>
      <asp:ListItem>Id</asp:ListItem>
      <asp:ListItem>Comment</asp:ListItem>
      </asp:DropDownList>
  </li>
  <li class="clearfix">
    <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-3 last"
      runat="server" Text="*ObjectValue :" />
    <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
      Text="1" />
  </li>
  <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
  <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxActivityList" runat="server" ItemPlaceholderID="aspItemPlaceholder"
  Visible="false">
  <EmptyDataTemplate>
    Criteria did not match any records.</EmptyDataTemplate>
  <LayoutTemplate>
    <table class="devsite-api-method">
      <thead>
        <tr>
          <th>Comment Id</th>
          <th>Activity Id</th>
          <th>User Id</th>
          <th>Comment</th>
        </tr>
      </thead>
      <tbody>
        <asp:PlaceHolder ID="aspItemPlaceholder" runat="server"></asp:PlaceHolder>
      </tbody>
    </table>
  </LayoutTemplate>
  <ItemTemplate>
    <tr>
      <td class="devsite-method"><%# Eval("Id")%></td>
      <td class="devsite-method"><%# Eval("ActivityId")%></td>
      <td class="devsite-method"><%# Eval("UserId")%></td>
      <td class="devsite-method"><%# Eval("Comment")%></td>
    </tr>
  </ItemTemplate>
</asp:ListView>
```

GetList .aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Common;
```

GetList .aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
{
  try
  {
    object Objectvalue;
    string Property = uxActivityProperty.SelectedItem.Text;
    ActivityCommentManager activityCommentManager = new ActivityCommentManager();
    ActivityCommentCriteria criteria = new ActivityCommentCriteria();
    criteria.OrderByField = ActivityCommentProperty.Id;
    criteria.OrderByDirection = EkEnumeration.OrderByDirection.Ascending;

    if (Property == "Id")
    {
      Objectvalue = Convert.ToInt64(uxObjectValue.Text);
      criteria.AddFilter(ActivityCommentProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
    }
    else if (Property == "UserId")
    {
      Objectvalue = Convert.ToInt64(uxObjectValue.Text);
      criteria.AddFilter(ActivityCommentProperty.UserId, CriteriaFilterOperator.EqualTo,
Objectvalue);
    }
    else
    {
      criteria.AddFilter(ActivityCommentProperty.Comment,
CriteriaFilterOperator.Contains, uxObjectValue.Text);
    }
    //Get Activity Comment List for given Filter
    List<ActivityCommentData> activityCommentList = activityCommentManager.GetList
(criteria);

    uxActivityList.Visible = true;
    uxActivityList.DataSource = activityCommentList;
    uxActivityList.DataBind();

    uxPageMultiView.SetActiveView(uxViewMessage);
  }
  catch (Exception ex)
  {
    MessageUtilities.UpdateMessage(uxMessage, ex.Message, Message.DisplayModes.Error);
    uxPageMultiView.SetActiveView(uxViewMessage);
  }
}
```

Update

```
Update(Ektron.Cms.ActivityCommentData)
```

Updates an existing ActivityCommentData item in the CMS.

This method takes one argument for the ActivityCommentData data class. When you use the ActivityCommentData, the method returns an updated custom ActivityCommentData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity Comment ID
- Activity ID
- Activity Comment

Parameters

- `comment`. Activity comment to update.

Returns

Returns the custom CmsData object updated.

Remarks

Validate the ActivityComment item with `GetItem()` before you update the properties. Then, call `Update` with your modified ActivityCommentData object.

NOTE: You cannot change all properties after the initial Add event, such as the ActivityCommentData.Type.

Update .aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxActivityCommentIDLabel"
AssociatedControlID="uxActivityCommentID" CssClass="span-4 last"
      runat="server" Text="*ActivityComment ID :" />
    <ektronUI:TextField ID="uxActivityCommentID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
      Text="1" />
  </li>
  <li class="clearfix">
    <ektronUI:Label ID="uxActivityIDLabel" AssociatedControlID="uxActivityID"
CssClass="span-4 last"
      runat="server" Text="Activity ID :" />
    <ektronUI:TextField ID="uxActivityID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
      Text="0" />
```

```
  </li>
  <li class="clearfix">
    <ektronUI:Label ID="uxActivityCommentLabel" AssociatedControlID="uxActivityComment"
      CssClass="span-4 last" runat="server" Text="Activity Comment :" />
    <ektronUI:TextField ID="uxActivityComment" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
      Text="" />
  </li>
  <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
  <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

Update .aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

Update .aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
{
  try
  {
    //Check whether user logged in or not
    if (activityCommentManager.UserId > 0)
    {
      activityCommentID = long.Parse(uxActivityCommentID.Text);
      //Check, Entered  activityCommentID valid or Not
      activityCommentData = activityCommentManager.GetItem(activityCommentID);
      //Check the Activity Comment Data
      if (activityCommentData != null)
      {
        ActivityManager activityManager = new ActivityManager();
        long activityID = Convert.ToInt64(uxActivityID.Text);
        if (activityID > 0)
       {
          //Check the ActivityDetails
          ActivityData activityData = activityManager.GetItem(activityID);
          if (activityData != null)
          {
            //Update the Activity Comment Details
            activityCommentData.ActivityId = activityID;
            update();
          }
          else
```

```
          {
            MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity ID and
Try again.", Message.DisplayModes.Error);
          }
        }
        else
        {
          update();
        }

      }
        else { MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
Comment ID.", Message.DisplayModes.Error); }
    }
    else
    {  MessageUtilities.UpdateMessage(uxMessage, "Please Login and try again.",
Message.DisplayModes.Error); }
    uxPageMultiView.SetActiveView(uxViewMessage);
  }
  catch (Exception ex)
  {
    MessageUtilities.UpdateMessage(uxMessage, ex.Message, Message.DisplayModes.Error);
    uxPageMultiView.SetActiveView(uxViewMessage);
  }
}
```

Data Classes

ActivityCommentCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Activity
```

Constructors

- `ActivityCommentCriteria()`

  ```
  public ActivityCommentCriteria()
  ```

- `ActivityCommentCriteria (Ektron.Cms.Activity.ActivityCommentProperty, EkEnumeration.OrderByDirection)`

  ```
  public ActivityCommentCriteria(Ektron.Cms.Activity.ActivityCommentProperty
  orderByField,
    EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `ActivityCommentProperty` are:

  - ActivityId
  - CanDelete
  - Comment
  - DateCreated

- Id
- UserAvatar
- UserDisplayName
- UserId

Properties

- `ReturnCommentCount`. Gets or sets the number of comments to return for each activity in the requested activity list. If 0, no comments will be returned. If -1, All Comments will be returned.

```
public int ReturnCommentCount { set; get; }
```

ActivityCommentData

Namespace

```
Ektron.Cms.Activity
```

Properties

- `ActivityId`. Gets or sets the Id of the Activity the comment belongs to.

```
public long ActivityId { set; get; }
```

- `CanDelete`. Gets or sets the Can delete permissions for the comment.

```
public bool CanDelete { set; get; }
```

- `Comment`. Gets or sets the comment text.

```
public string Comment { set; get; }
```

- `DateCreated`. Gets or sets the date the comment was created.

```
public System.DateTime DateCreated { set; get; }
```

- `Id`. Gets or sets the Id of the comment.

```
public long Id { set; get; }
```

- `TimeLapse`. Gets or sets the time lapse for the comment.

```
public string TimeLapse { set; get; }
```

- `UserAvatar`. Gets or sets the avatar of the user who made the comment.

```
public string UserAvatar { set; get; }
```

- `UserDisplayName`. Gets or sets the display name of the user who made the comment.

```
public string UserDisplayName { set; get; }
```

- `UserId`. Gets or sets the ID of the user who made the comment.

```
Gets or sets the display name of the user who made the comment.
```

ActivityManager

8.50 and higher

The ActivityManager class manages user activity items that appear in the activity stream.

Namespace

```
Ektron.Cms.Framework.Activity.ActivityManager
```

Constructors

- `ActivityManager()`
- `ActivityManager(ApiAccessMode)`

Properties

- `ActivityService`. Gets instance to IActivityStream.
- `ActivityStreamService`. Gets instance to IActivityStream.
- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsActivityEmailReplyEnabled`. Determines whether Activity Email Replies is enabled. Use `EnableActivityEmailReply` and `DisableActivityEmailReply` to set the property.
- `IsActivityPublishingEnabled`. Determines whether Activity Publishing is enabled. Use `EnableActivityPublishing` and `DisableActivityPublishing` to set the property.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *DisableActivityEmailReply* on page 183
- *DisableActivityPublishing* on page 184
- *EnableActivityEmailReply* on page 186
- *EnableActivityPublishing* on page 187
- *GetItem* on page 189
- *GetList* on page 191
- *GetListForUser (1)* on page 197
- *GetListForUser (2)* on page 199
- *Publish* on page 202
- `PurgeActivity(DateTime)`. Purges the activity older than given date.
- *Update* on page 203
- *UpdateActivityMessage* on page 206

Add

```
Add(Ektron.Cms.Activity.ActivityData)
```

Adds Activity data to the activity stream.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Event ID
- * Activity Type ID

Parameters

- `comment`. Activity data to add.

Returns

- Returns the custom CmsData object added.

Remarks

This method takes one argument for the ActivityData class. When you add the Activity data, the method returns a new custom ActivityData object.

Set a valid ActvityTypeID based on the activity item or the ObjectID. To get available activity types, use `ActivityTypeManager. GetList (ActivityTypeCriteria)` or choose **Workarea > Settings > Community Management > Notifications > ActivityTypes**.

Add .aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxWebEventIDLabel" AssociatedControlID="uxWebEventID"
CssClass="span-4 last"
      runat="server" Text="* WebEventID :" />
    <ektronUI:TextField ID="uxWebEventID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
      Text="251" />
  </li>
  <li class="clearfix">
    <ektronUI:Label ID="uxMesageLabel" AssociatedControlID="uxActivityMessage"
CssClass="span-4 last"
      runat="server" Text="*Activity Mesage :" />
    <ektronUI:TextField ID="uxActivityMessage" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
      Text="A NewCalendarEvent is added" />
  </li>
  <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
  <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

Add .aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Common.Calendar;
using Ektron.Cms.Framework.Calendar;Add .aspx.cs
```

Code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityManager activityManager = new ActivityManager();
            WebEventManager webeventManager = new WebEventManager();
            long eventid = Convert.ToInt64(uxWebEventID.Text);
            //Check the WebEvent Details
            WebEventData WebeventData = webeventManager.GetItem(eventid);
            if (WebeventData != null)
            {
                ActivityUserInfo userInfo = new ActivityUserInfo();
                userInfo.Id = activityManager.UserId;
```

```
                //Set the Activity Details
                ActivityData activityData = new ActivityData()
                {
                    ActivityTypeId = 25,
                    ActionUser = userInfo,
                    Message = uxActivityMessage.Text,
                    ObjectId = eventid,
                    Date = DateTime.Now,
                    LanguageId = 1033
                };
                //Add a New Activity with given activityDetails.
                activityManager.Add(activityData);
                MessageUtilities.UpdateMessage(uxMessage, "Activity Saved with Id " +
activityData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid WebEvent
ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

AddActivityMessage

```
AddActivityMessage(System.Int64,System.Int32,System.String)
```

Adds a language-specific message for an activity to the activity stream.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity ID
- * Language ID
- * Activity Message

Parameters

- `activityId`. ID of activity to add message for.
- `languageId`. ID of language to add message for.

- `message`. Activity message to be added.

Code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityIDLabel" AssociatedControlID="uxActivityID"
CssClass="span-4 last"
            runat="server" Text="*Activity ID :" />
        <ektronUI:TextField ID="uxActivityID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageLabel" AssociatedControlID="uxLanguageId"
CssClass="span-4 last"
            runat="server" Text="*Language ID :" />
        <ektronUI:TextField ID="uxLanguageId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1033" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMesageLabel" AssociatedControlID="uxActivityMessage"
CssClass="span-4 last"
            runat="server" Text="*Activity Message :" />
        <ektronUI:TextField ID="uxActivityMessage" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

Code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Common.Calendar;
using Ektron.Cms.Framework.Calendar;
```

Code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            ActivityManager activityManager = new ActivityManager();
            long activityID = Convert.ToInt64(uxActivityID.Text);
            //Check the ActivityDetails
            ActivityData activityData = activityManager.GetItem(activityID);
            if (activityData != null)
            {
                //Add a Activity with given activityDetails.
                activityManager.AddActivityMessage(activityID, Convert.ToInt32
(uxLanguageId.Text), uxActivityMessage.Text);
                MessageUtilities.UpdateMessage(uxMessage, "Activity Saved with Id " +
activityData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Delete

```
Delete(System.Int64)
```

Deletes an activity from the activity stream.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity ID

Parameters

- `comment`. Activity data to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityIDLabel" AssociatedControlID="uxActivityID"
CssClass="span-4 last"
            runat="server" Text="*Activity ID :" />
        <ektronUI:TextField ID="uxActivityID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityManager activityManager = new ActivityManager();
            long activityID = Convert.ToInt64(uxActivityID.Text);
            //Check the ActivityDetails
            ActivityData activityData = activityManager.GetItem(activityID);
            if (activityData != null)
            {   //Delete the  Activity ID from the CMS
                activityManager.Delete(activityID);
                MessageUtilities.UpdateMessage(uxMessage, "Activity  with Id " +
activityData.Id + " has been Deleted", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
```

```
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

DisableActivityEmailReply

```
DisableActivityEmailReply(ActivityData)
```

Disables activity email replies for the site.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Disable Activity Email Reply

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxDisableActivityEmailLabel"
AssociatedControlID="uxDisableActivityEmail"
            CssClass="span-5 last" runat="server" Text="*Disable Activity Email Reply:"
/>
        <ektronUI:Button ID="uxDisableActivityEmail" CssClass="span-5"
            runat="server" ValidationGroup="RegisterValidationGroup"
            DisplayMode="Checkbox" Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
            Text="Disable"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4"
                runat="server" Text="* - Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server">
</asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
```

```
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityManager activityManager = new ActivityManager();
            if (!activityManager.IsActivityEmailReplyEnabled)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Activity Email Reply is
Already Disabled.", Message.DisplayModes.Success);
            }
            else
            {
                if (uxDisableActivityEmail.Checked)
                {   //Disable the Activity Email Reply
                    activityManager.DisableActivityEmailReply();
                    MessageUtilities.UpdateMessage(uxMessage, "Activity Email Reply has
been Disabled.", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Activity Email Reply is
Not Disabled.", Message.DisplayModes.Error);
                }
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

DisableActivityPublishing

```
DisableActivityPublishing(Ektron.Cms.Activity.ActivityData)
```

Disables activity publishing for the site.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Disable Activity Publishing

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxDisableActivityPublishLabel"
AssociatedControlID="uxDisableActivityPublish"
            CssClass="span-4 last" runat="server" Text="*Disable Activity Publish :" />
        <ektronUI:Button ID="uxDisableActivityPublish" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            DisplayMode="Checkbox" Text=""/>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Disable"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityManager activityManager = new ActivityManager();
            if (!activityManager.IsActivityPublishingEnabled)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Activity Publishing is
Already Disabled.", Message.DisplayModes.Success);
            }
            else
            {
                if (uxDisableActivityPublish.Checked)
                {   //Disable the Activity Publishing
                    activityManager.DisableActivityPublishing();
                    MessageUtilities.UpdateMessage(uxMessage, "Activity Publishing has
been Disabled.", Message.DisplayModes.Success);
                }
                else
                {
```

```
                    MessageUtilities.UpdateMessage(uxMessage, "Activity Publishing is
Not Disabled.", Message.DisplayModes.Error);
                }
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

EnableActivityEmailReply

```
EnableActivityEmailReply(Ektron.Cms.Activity.ActivityData)
```

Enables activity email replies for the site.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Enable Activity Email Reply

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxEnableActivityEmailLabel"
AssociatedControlID="uxEnableActivityEmail"
                CssClass="span-4 last" runat="server" Text="*Enable Activity Email Reply
:" />
            <ektronUI:Button ID="uxEnableActivityEmail" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
                    DisplayMode="Checkbox" />
        </li>
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Enable"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityManager activityManager = new ActivityManager();
            if (activityManager.IsActivityEmailReplyEnabled)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Activity Email Reply is
Already Enabled.", Message.DisplayModes.Success);
            }
            else
            {
                if (uxEnableActivityEmail.Checked)
                {   //Enable the Activity Email Reply
                    activityManager.EnableActivityEmailReply();
                    MessageUtilities.UpdateMessage(uxMessage, "Activity Email Reply has
been Enabled.", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Activity Email Reply is
Not Enabled.", Message.DisplayModes.Error);
                }
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

EnableActivityPublishing

```
EnableActivityPublishing(Ektron.Cms.Activity.ActivityData)
```

Enables activity publishing for the site and set up default notification preferences for existing users.

NOTE: Depending on the number of users, this call could be time-consuming.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Enable Activity Publishing

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEnableActivityPublishLabel"
AssociatedControlID="uxEnableActivityPublish"
            CssClass="span-4 last" runat="server" Text="*EnableActivity Publish :" />
        <ektronUI:Button ID="uxEnableActivityPublish" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
                DisplayMode="Checkbox" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Enable"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityManager activityManager = new ActivityManager();
            if (activityManager.IsActivityPublishingEnabled)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Activity Publishing is
Already Enabled.", Message.DisplayModes.Success);
            }
```

```
            else
            {
                if (uxEnableActivityPublish.Checked)
                {   //Enable the Activity Publishing
                    activityManager.EnableActivityPublishing();
                    MessageUtilities.UpdateMessage(uxMessage, "Activity Publishing has
been Enabled.", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage,"Activity Publishing is Not
Enabled.", Message.DisplayModes.Warning);
                }
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetItem

```
GetItem(System.Int64)
```

Returns a single ActivityStream Item by ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity ID

Parameters

- `id`. ID of activity to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityIDLabel" AssociatedControlID="uxActivityID"
CssClass="span-4 last"
            runat="server" Text="*Activity ID :" />
        <ektronUI:TextField ID="uxActivityID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
```

```
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxObjectId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxActionUserID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxActivityMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxActivityTypeID" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityManager activityManager = new ActivityManager();
            long activityID = Convert.ToInt64(uxActivityID.Text);
            //Check the ActivityDetails
            ActivityData activityData = activityManager.GetItem(activityID);
            if (activityData != null)
            {   //Display the Activity Details

                MessageUtilities.UpdateMessage(uxMessage, "Activity Details  for
ActivityId " + activityData.Id + " are below", Message.DisplayModes.Success);
                uxObjectId.Text = "Object ID: " + activityData.ObjectId.ToString();
                uxActionUserID.Text = "Action UserID: " +
activityData.ActionUser.Id.ToString();
```

```
                uxActivityMessage.Text = "Activity Message: " + activityData.Message;
                uxActivityTypeID.Text = "Activity Type ID: " +
activityData.ActivityTypeId.ToString();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetList

```
GetList(Ektron.Cms.Activity.ActivityCriteria
  Ektron.Cms.Common.EkEnumeration.ActivityFeedType,System.Int64)
```

Gets an Activity Feed based upon the feedtype and the ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Activity Property
- * Object Value

Parameters

- `criteria`. ActivityCriteria used to filter results.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityPropertyLabel"
AssociatedControlID="uxActivityProperty"
            CssClass="span-3 last" runat="server" Text="Feed Type :" />
        <asp:DropDownList ID="uxActivityProperty" runat="server">
            <asp:ListItem>User</asp:ListItem>
            <asp:ListItem>CommunityGroup</asp:ListItem>
        </asp:DropDownList>
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-3 last"
            runat="server" Text="*ID :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxActivityList" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        ObjectId
                    </th>
                    <th>
                        ActivityTypeId
                    </th>
                    <th>
                        Message
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ActivityTypeId")%>
            </td>
            <td class="devsite-method">
```

```
                <%# Eval("Message")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long feedTypeId;
            string Property = uxActivityProperty.SelectedItem.Text;

            ActivityManager activityManager = new ActivityManager();
            ActivityCriteria criteria = new ActivityCriteria();
            List<ActivityData> activityList = new List<ActivityData>();

            feedTypeId = Convert.ToInt64(uxObjectValue.Text);
            if (Property == "User")
            {
                //Get Activity List for given feedTypeId
                activityList = activityManager.GetList(criteria,
Ektron.Cms.Common.EkEnumeration.ActivityFeedType.User, feedTypeId);
            }
            else
            {
                //Get Activity List for given feedTypeId
                activityList = activityManager.GetList(criteria,
Ektron.Cms.Common.EkEnumeration.ActivityFeedType.CommunityGroup, feedTypeId);
            }

            uxActivityList.Visible = true;
            uxActivityList.DataSource = activityList;
            uxActivityList.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetList

```
GetList
(Ektron.Cms.Activity.ActivityCriteria,Ektron.Cms.Common.EkEnumeration.ActivityFeedType,S
ystem.Int64)
```

Gets an Activity Feed based upon the feed type and the ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Feed Type
- * ID

Parameters

- `criteria`. ActivityCriteria for further filtering the feed, sorting, and paging.
- `feedType`. The type of feed to retrieve. For example, User or CommunityGroup.
- `feedTypeId`. The ID corresponding to the feed type. For example, if FeedType = User, FeedTypeId would be the user ID to retrieve feed for.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityPropertyLabel"
AssociatedControlID="uxActivityProperty" CssClass="span-3 last"
            runat="server" Text="Activity Property :" />
        <asp:DropDownList ID="uxActivityProperty" runat="server">
            <asp:ListItem>ActionUserId</asp:ListItem>
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>ActivityTypeId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-3 last"
            runat="server" Text="*ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
```

```
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxActivityList" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        ObjectId
                    </th>
                    <th>
                        ActivityTypeId
                    </th>
                    <th>
                        Message
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ActivityTypeId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Message")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            string Property = uxActivityProperty.SelectedItem.Text;
            ActivityManager activityManager = new ActivityManager();
            ActivityCriteria activityCriteria = new ActivityCriteria();
            Objectvalue = Convert.ToInt64(uxObjectValue.Text);
            if (Property == "Id")
            {
                activityCriteria.AddFilter(ActivityProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (Property == "ActivityTypeId")
            {
                activityCriteria.AddFilter(ActivityProperty.ActivityTypeId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                activityCriteria.AddFilter(ActivityProperty.ActionUserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            //Get Activity List for given Filter
            List<ActivityData> activityList = activityManager.GetList(activityCriteria);

            uxActivityList.Visible = true;
            uxActivityList.DataSource = activityList;
            uxActivityList.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

```
    }
```

GetListForUser (1)

```
GetListForUser(System.Int64,Ektron.Cms.PagingInfo)
```

Retrieves a list of activities associated with a user that is the subject or the actor.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `pagingInfo`. Paging information for retrieval.
- `userId`. ID of user to retrieve feed for.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxActivityList" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
```

```
                    <th>
                        ObjectId
                    </th>
                    <th>
                        ActivityTypeId
                    </th>
                    <th>
                        Message
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ActivityTypeId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Message")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityManager activityManager = new ActivityManager();
```

```
            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxUserId.Text);
            //Check the given UserID valid or Not
            UserData Userdata = Usermanager.GetItem(UserId);
            if (Userdata != null)
            {    //Paging Information
                PagingInfo paging = new PagingInfo();
                paging.CurrentPage = 1;
                paging.RecordsPerPage = 50;
                //Get the Activity List
                List<ActivityData> activityList = activityManager.GetListForUser(UserId,
paging);
                //Display the Activity List
                uxActivityList.Visible = true;
                uxActivityList.DataSource = activityList;
                uxActivityList.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid User ID
and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetListForUser (2)

```
GetListForUser(System.Int64,Ektron.Cms.PagingInfo,System.Int32)
```

Retrieves a list of activities associated with a user that is the subject or the Actor.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- Return Comment Count

Parameters

- `pagingInfo`. Paging information for retrieval.
- `userId`. ID of user to retrieve feed for.

- `returnCommentCount`. Number of comments to return with each activity. 0 = no comments, -1 = all.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-5 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCommentCountLabel" AssociatedControlID="uxCommentCount"
CssClass="span-5 last"
            runat="server" Text="ReturnComment Count:" />
        <asp:DropDownList ID="uxCommentCount" runat="server">
            <asp:ListItem>-1</asp:ListItem>
            <asp:ListItem>0</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxActivityList" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        ObjectId
                    </th>
                    <th>
                        ActivityTypeId
                    </th>
                    <th>
                        Message
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
```

```
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ActivityTypeId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Message")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityManager activityManager = new ActivityManager();
            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxUserId.Text);
            //Check the given UserID valid or Not
            UserData Userdata = Usermanager.GetItem(UserId);
            if (Userdata != null)
            {    //Paging Information
                PagingInfo paging = new PagingInfo();
                paging.CurrentPage = 1;
                paging.RecordsPerPage = 50;
                //Return Comment Count value i.e -1 (ALL comments) or 0
                int returnCommentCount = int.Parse(uxCommentCount.SelectedItem.Text);
                //Get the Activity List with Comments
                List<ActivityData> activityList = activityManager.GetListForUser(UserId,
```

```
paging, returnCommentCount);
                //Display the Activity List
                uxActivityList.Visible = true;
                uxActivityList.DataSource = activityList;
                uxActivityList.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid User ID
and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Publish

```
Publish(Ektron.Cms.Activity.ActivityData)
```

Publishes an activity to all subscribers.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- *Activity ID

Parameters

- `activity`. ActivityData that has occurred.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityIDLabel" AssociatedControlID="uxActivityID"
CssClass="span-4 last"
            runat="server" Text="*Activity ID :" />
        <ektronUI:TextField ID="uxActivityID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
```

```
Text="Publish"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityManager activityManager = new ActivityManager();
            long activityID = Convert.ToInt64(uxActivityID.Text);
            //Check the ActivityDetails
            ActivityData activityData = activityManager.GetItem(activityID);
            if (activityData != null)
            {
                //Publish Activity with given activityDetails.
                activityManager.Publish(activityData);
                 MessageUtilities.UpdateMessage(uxMessage, "Activity with Id " +
activityData.Id + " has been Published.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Please Enter Valid Activity
ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Update

```
Update(Ektron.Cms.ActivityData)
```

Updates an existing Activity item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity ID
- Web Event ID
- Activity Message

Parameters

- `activity`. ActivityData to update.

Remarks

Validate the Activity item with `GetItem()` before you update the properties. Then, call `Update` with your modified ActivityData object.

NOTE: You cannot change all properties after the initial Add event, such as the ActivityData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityIDLabel" AssociatedControlID="uxActivityID"
CssClass="span-4 last"
            runat="server" Text="*Activity ID :" />
        <ektronUI:TextField ID="uxActivityID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxWebEventIDLabel" AssociatedControlID="uxWebEventID"
CssClass="span-4 last"
            runat="server" Text="WebEventID :" />
        <ektronUI:TextField ID="uxWebEventID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMesageLabel" AssociatedControlID="uxActivityMessage"
CssClass="span-4 last"
            runat="server" Text="Activity Mesage :" />
        <ektronUI:TextField ID="uxActivityMessage" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
```

```
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Common.Calendar;
using Ektron.Cms.Framework.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityManager activityManager = new ActivityManager();
            long activityID = Convert.ToInt64(uxActivityID.Text);
            //Check the ActivityDetails
            ActivityData activityData = activityManager.GetItem(activityID);
            if (activityData != null)
            {
                if (activityData.ActivityTypeId == 25)
                {
                    WebEventManager webeventManager = new WebEventManager();
                    long eventid = Convert.ToInt64(uxWebEventID.Text);
                    if (eventid > 0)
                    {
                        //Check the WebEvent Details
                        WebEventData WebeventData = webeventManager.GetItem(eventid);
                        if (WebeventData != null)
                        {
                            activityData.ObjectId = eventid;
                        }
                    }
                    else
                    {
                        activityData.ObjectId = activityData.ObjectId;
                    }
                    activityData.Message = uxActivityMessage.Text != string.Empty ?
uxActivityMessage.Text : activityData.Message;
                    //Update Activity with given activity Details.
```

```
                    activityManager.Update(activityData);

                    MessageUtilities.UpdateMessage(uxMessage, "Activity  with Id " +
activityData.Id + " has been Updated", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please Enter a WebEvent
Activity ID and try again", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

UpdateActivityMessage

```
UpdateActivityMessage(System.Int64,System.Int32,System.String)
```

Updates an existing message for an activity to the activity stream.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity ID
- Language ID
- Activity Message

Parameters

- `activityId`. ID of activity to add message for.
- `languageId`. ID of language to add message for.
- `message`. Activity message to be added.

.aspx code snippet

```
ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityIDLabel" AssociatedControlID="uxActivityID"
CssClass="span-4 last"
            runat="server" Text="*Activity ID :" />
        <ektronUI:TextField ID="uxActivityID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageLabel" AssociatedControlID="uxLanguageId"
CssClass="span-4 last"
            runat="server" Text="Language ID :" />
        <ektronUI:TextField ID="uxLanguageId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1033" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMesageLabel" AssociatedControlID="uxActivityMessage"
CssClass="span-4 last"
            runat="server" Text="Activity Message :" />
        <ektronUI:TextField ID="uxActivityMessage" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Common.Calendar;
using Ektron.Cms.Framework.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            ActivityManager activityManager = new ActivityManager();
            long activityID = Convert.ToInt64(uxActivityID.Text);
            //Check the ActivityDetails
            ActivityData activityData = activityManager.GetItem(activityID);
            if (activityData != null)
            {
                string message = uxActivityMessage.Text != string.Empty ?
uxActivityMessage.Text : activityData.Message;
                activityManager.UpdateActivityMessage(activityID, Convert.ToInt32
(uxLanguageId.Text), message);
                //Update a Activity with given activityDetails.
                MessageUtilities.UpdateMessage(uxMessage, "Activity Details for Id " +
activityData.Id + " Has been Updated", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Data Classes

ActivityCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Activity
```

Constructors

- `ActivityCriteria()`

  ```
  public ActivityCriteria()
  ```

- `ActivityCriteria(Ektron.Cms.Activity.ActivityProperty, EkEnumeration.OrderByDirection)`

  ```
  public ActivityCriteria(Ektron.Cms.Activity.ActivityProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `ActivityProperty` are:

  - ActionType
  - ActionUserDisplayName
  - ActionUserId

- ActivityTypeId
- CanComment
- CommentCount
- Date
- Id
- LanguageId
- Message
- ObjectId
- ObjectType
- ScopeObjectId
- ScopeObectType
- SiteId
- UniqueId

- `ReturnCommentCount`. Gets or sets the number of comments to return for each activity in the requested activity list. If 0, no comments are returned. If -1, all comments are returned.

```
public int ReturnCommentCount { set; get; }
```

ActivityData

Namespace

```
Ektron.Cms.Activity
```

Properties

- `ActionUser`. Gets or sets the user who caused this activity to occur. For example, the user who posted a blog post or uploaded a document.

```
public Ektron.Cms.Activity.ActivityUserInfo ActionUser { set; get; }
```

- `ActivityData()`

```
public ActivityData()
```

- `ActivityData(long, long, long)`

```
public ActivityData(long activityTypeId, long objectId,
  long actionUserId)
```

- `ActivityData(long, long, long, int, string)`

```
public ActivityData(long activityTypeId, long objectId,
  long actionUserId, int languageId, string message)
```

- `ActivityTypeId`. Gets or sets the ActivityType for this Activity. Built in CMS Activity are defined in `EkEnumeration.ActivityType`.

```
public long ActivityTypeId { set; get; }
```

- `CanComment`. Gets or sets the CanComment permissions for the activity.

```
public bool CanComment { set; get; }
```

- `CommentCount`. Gets or sets the Comment count for the activity.

```
public int CommentCount { set; get; }
```

- `Comments`. Gets or sets the Comments for the activity. This property will not be populated unless explicitly specified when calling API. To get the total comment count, see `CommentCount` property.

```
public System.Collections.ObjectModel.Collection<ActivityCommentData>
  Comments { set; get; }
```

- `Date`. Gets or sets the date the activity occurred.

```
public System.DateTime Date { set; get; }
```

- `Id`. Gets or set the ID of the activity.

```
public long Id { set; get; }
```

- `LanguageId`. Gets or sets the language associated with this activity's message.

```
public int LanguageId { set; get; }
```

- `Message`. Gets or sets the message for this activity.

```
public string Message { set; get; }
```

- `ObjectId`. Gets or sets the ID of the object associated with this activity.

```
public long ObjectId { set; get; }
```

- `ScopeObjectId`. Gets or sets the ID of the scope object associated with this activity.

```
public long ScopeObjectId { set; get; }
```

- `SiteId`. Gets or sets the ID of the site that the activity originated from.

```
public long SiteId { set; get; }
```

- `TimeLapse`. Gets the time lapse for the activity.

```
public string TimeLapse { set; get; }
```

- `UniqueId`. Gets or sets the unique ID for the activity.

```
public System.Guid UniqueId { set; get; }
```

- `Validate()`. Validates the current ActivityData instance is in a valid state for saving. Returns collection of validation errors if not valid.

```
public ValidationResults Validate()
```

ActivityStreamManager

8.50 and higher

The ActivityStreamManager class manages gets post from and sends posts to the activity stream.

Namespace

```
Ektron.Cms.Framework.Activity.ActivityStreamManager
```

Constructors

- `ActivityStreamManager()`
- `ActivityStreamManager(ApiAccessMode)`

Properties

- `ActivityStreamService`. Gets instance to IActivityStream.
- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

GetUserActivityStream (user activity)

```
GetUserActivityStream(System.Int64,Ektron.Cms.PagingInfo)
```

Retrieves a list of activities in a user's activity stream.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `paging`. Paging information for retrieval.
- `userId`. ID of user to retrieve feed for.

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxActivityList" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        ObjectId
                    </th>
                    <th>
                        ActivityTypeId
                    </th>
                    <th>
                        Message
                    </th>
                </tr>
```

```
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ActivityTypeId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Message")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityStreamManager activityStreamManager = new ActivityStreamManager();
            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxUserId.Text);
            //Check the given UserID valid or Not
            UserData Userdata = Usermanager.GetItem(UserId);
            if (Userdata != null)
            {    //Paging Information
                PagingInfo paging = new PagingInfo();
                paging.CurrentPage = 1;
                paging.RecordsPerPage = 50;
                //Get the Activity List
```

```
                List<ActivityData> activityList =
activityStreamManager.GetUserActivityStream(UserId, paging);
                //Display the Activity List
                uxActivityList.Visible = true;
                uxActivityList.DataSource = activityList;
                uxActivityList.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid User ID
and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetUserActivityStream (specific user activity)

```
GetUserActivityStream(Ektron.Cms.Activity.ActivityCriteria,System.Int64)
```

Retrieves a list of activities in a user's activity stream based upon supplied criteria.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object Value
- * User ID

Parameters

- `criteria`. Criteira by which to filter the stream. UserId filter is ignored.
- `userId`. ID of user to retrieve feed for.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityPropertyLabel"
AssociatedControlID="uxActivityProperty"
            CssClass="span-3 last" runat="server" Text="Activity Property :" />
        <asp:DropDownList ID="uxActivityProperty" runat="server">
            <asp:ListItem>ActivityTypeId</asp:ListItem>
```

```
            <asp:ListItem>Id</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-3 last"
            runat="server" Text="*ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="20" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxActivityList" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        ObjectId
                    </th>
                    <th>
                        ActivityTypeId
                    </th>
                    <th>
                        Message
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
```

```
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ActivityTypeId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Message")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            string Property = uxActivityProperty.SelectedItem.Text;
            long UserId = long.Parse(uxUserId.Text);

            ActivityStreamManager activityStreamManager = new ActivityStreamManager();
            ActivityCriteria criteria = new ActivityCriteria();
            criteria.OrderByField = ActivityProperty.Id;
            criteria.OrderByDirection = EkEnumeration.OrderByDirection.Ascending;

            UserManager Usermanager = new UserManager();

            //Check the given UserID valid or Not
            UserData Userdata = Usermanager.GetItem(UserId);
            if (Userdata != null)
```

```
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                if (Property == "Id")
                {
                    criteria.AddFilter(ActivityProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
                }
                else
                {
                    criteria.AddFilter(ActivityProperty.ActivityTypeId,
CriteriaFilterOperator.EqualTo, Objectvalue);
                }
                //Get the Activity List
                List<ActivityData> activityList =
activityStreamManager.GetUserActivityStream(criteria, UserId);
                //Display the Activity List
                uxActivityList.Visible = true;
                uxActivityList.DataSource = activityList;
                uxActivityList.DataBind();
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User ID and
Try again.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

SendActivityToAllStreams

```
SendActivityToAllStreams(System.Int64)
```

Sends the activity to all user streams based upon the activity and users preferences.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity ID

Parameters

- `activityId`. ID of activity to send to streams.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityIDLabel" AssociatedControlID="uxActivityID"
CssClass="span-3 last"
            runat="server" Text="*Activity ID :" />
        <ektronUI:TextField ID="uxActivityID" CssClass="span-3" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Send"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long activityID = Convert.ToInt64(uxActivityID.Text);

            ActivityStreamManager activityStreamManager = new ActivityStreamManager();
            ActivityManager activityManager = new ActivityManager();

            //Check the ActivityDetails
            ActivityData activityData = activityManager.GetItem(activityID);
            if (activityData != null)
            {   //Send a given Activity to all user's streams
                activityStreamManager.SendActivityToAllStreams(activityID);
                MessageUtilities.UpdateMessage(uxMessage, "Activity ID " + activityID +
" has sent to all user's streams.", Message.DisplayModes.Success);
            }
            else
```

```
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

SendActivityToStreams

```
SendActivityToStreams(System.Int64)
```

Sends the activity to all applicable streams based upon the activity and users preferences.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity ID

Parameters

- `activityId`. ID of activity to send to streams.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityIDLabel" AssociatedControlID="uxActivityID"
CssClass="span-3 last"
            runat="server" Text="*Activity ID :" />
        <ektronUI:TextField ID="uxActivityID" CssClass="span-3" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
```

```
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Send"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long activityID = Convert.ToInt64(uxActivityID.Text);

            ActivityStreamManager activityStreamManager = new ActivityStreamManager();
            ActivityManager activityManager = new ActivityManager();

            //Check the ActivityDetails
            ActivityData activityData = activityManager.GetItem(activityID);
            if (activityData != null)
            {   //Send a given Activity to all applicable streams
                activityStreamManager.SendActivityToStreams(activityID);
                MessageUtilities.UpdateMessage(uxMessage, "Activity ID " + activityID +
" has sent to all applicable streams.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

```
    }
```

SendActivityToUsersStream

```
SendActivityToUsersStream(System.Int64,System.Int64)
```

Sends an activity that is in the general activity stream to a user's activity stream.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity ID
- * User ID

Parameters

- `activityId`. ID of activity to add to users feed.
- `userId`. ID of user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityIDLabel" AssociatedControlID="uxActivityID"
CssClass="span-3 last"
            runat="server" Text="*Activity ID :" />
        <ektronUI:TextField ID="uxActivityID" CssClass="span-3" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Send"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long UserId = long.Parse(uxUserId.Text);
            long activityID = Convert.ToInt64(uxActivityID.Text);

            ActivityStreamManager activityStreamManager = new ActivityStreamManager();
            ActivityManager activityManager = new ActivityManager();

            //Check the ActivityDetails
            ActivityData activityData = activityManager.GetItem(activityID);
            if (activityData != null)
            {
                UserManager Usermanager = new UserManager();
                //Check the given UserID valid or Not
                UserData Userdata = Usermanager.GetItem(UserId);
                if (Userdata != null)
                {   //Send Activity to a user's Stream
                    activityStreamManager.SendActivityToUsersStream(activityID, UserId);
                    MessageUtilities.UpdateMessage(uxMessage, "Activity ID " +
activityID + " has been added to a general feed of user ID " + UserId,
Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid User
ID and Try again.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
```

```
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Data Classes

ActivityCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Activity
```

Constructors

- `ActivityCriteria()`

  ```
  public ActivityCriteria()
  ```

- `ActivityCriteria(Ektron.Cms.Activity.ActivityProperty, EkEnumeration.OrderByDirection)`

  ```
  public ActivityCriteria(Ektron.Cms.Activity.ActivityProperty orderByField,
    EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `ActivityProperty` are:

  - ActionType
  - ActionUserDisplayName
  - ActionUserId
  - ActivityTypeId
  - CanComment
  - CommentCount
  - Date
  - Id
  - LanguageId
  - Message
  - ObjectId
  - ObjectType
  - ScopeObjectId
  - ScopeObectType
  - SiteId
  - UniqueId

- `ReturnCommentCount`. Gets or sets the number of comments to return for each activity in the requested activity list. If 0, no comments are returned. If -1, all

comments are returned.

```
public int ReturnCommentCount { set; get; }
```

ActivityTypeManager

8.50 and higher

The ActivityTypeManager class manages activity types that appear in the activity stream.

Namespace

```
Ektron.Cms.Framework.Activity.ActivityTypeManager
```

Constructors

- `ActivityTypeManager()`
- `ActivityTypeManager(ApiAccessMode)`

Properties

- `ActivityTypeService`. Activity Type Service
- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 227
- *GetItem* on page 229
- *GetList* on page 231
- *Update* on page 234

Add

```
Add(Ektron.Cms.Activity.ActivityTypeData)
```

Adds an ActivityType to the activity stream.

This method takes one argument for the ActivityTypeData class. When you add the Activity Comment data, the method returns a new ActivityType.ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Name

Parameters

- `ActivityTypeData`. ActivityTypeData object to add.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-4
last"
            runat="server" Text="*Name :" />
        <ektronUI:TextField ID="uxName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityTypeManager activityTypeManager = new ActivityTypeManager();
            //Set the Activity Type Details
            ActivityTypeData activityTypeData = new ActivityTypeData()
            {
                Name = uxName.Text,
                ObjectType = Ektron.Cms.Common.EkEnumeration.ActivityObjectType.Content,
                ActionType = Ektron.Cms.Common.EkEnumeration.ActivityActionType.Add,
                Scope = Ektron.Cms.Common.EkEnumeration.ActivityScope.User,
                DisplayInPreferences = true,
                IsPublishByDefault = true,
                SupportsComments = true
            };
            //Add a New Activity Type with given activityTypeDetails.
            activityTypeManager.Add(activityTypeData);

            MessageUtilities.UpdateMessage(uxMessage, "Activity Type Saved with Id " +
activityTypeData.Id, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Delete

```
Delete(System.Int64)
```

Deletes an activity type from the activity stream.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity Type ID

Remarks

Validate the item using `GetItem()` before deleting.

Parameters

- `id`. ID of ActivityType to delete.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityTypeIDLabel"
AssociatedControlID="uxActivityTypeID"
            CssClass="span-4 last" runat="server" Text="*Activity TypeID :" />
        <ektronUI:TextField ID="uxActivityTypeID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>

    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityTypeManager activityTypeManager = new ActivityTypeManager();
            long activityTypeID = long.Parse(uxActivityTypeID.Text);
            //Check whether Entered activityTypeID valid or Not
            ActivityTypeData activityTypeData = activityTypeManager.GetItem
(activityTypeID);
            if (activityTypeData != null)
            {   //Delete  Activity Type ID from the CMS
                activityTypeManager.Delete(activityTypeID);
                MessageUtilities.UpdateMessage(uxMessage,  "Activity Type Id " +
activityTypeID + " has been Deleted from the CMS.", Message.DisplayModes.Success);
            }
            else
            {
```

```
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
Type ID", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific ActivityType object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity Type ID

Parameters

- `id`. ID of ActivityType to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityTypeIDLabel"
AssociatedControlID="uxActivityTypeID"
            CssClass="span-4 last" runat="server" Text="*Activity TypeID :" />
        <ektronUI:TextField ID="uxActivityTypeID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
```

```
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxObjectType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSupportsComments" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityTypeManager activityTypeManager = new ActivityTypeManager();
            long activityTypeID = long.Parse(uxActivityTypeID.Text);
            //Check whether Entered activityTypeID valid or Not
            ActivityTypeData activityTypeData = activityTypeManager.GetItem
(activityTypeID);
            if (activityTypeData != null)
            {   //Display given activity TypeId Details
                MessageUtilities.UpdateMessage(uxMessage,  "Activity Type Details for ID
" + activityTypeID + " are below", Message.DisplayModes.Success);
                uxName.Text = "Activity Type Name = " + activityTypeData.Name;
                uxObjectType.Text = "Object Type = " + activityTypeData.ObjectType;
                uxSupportsComments.Text = "Supports Comments = " +
activityTypeData.SupportsComments;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
Type ID", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }

    }
```

GetList

```
GetList(Ektron.Cms.Framework.Activity.ActivityTypeCriteria)
```

Retrieves lists of objects through the GetList(ActivityTypeCriteria) method.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Activity Property
- * Object Value

Parameters

- `criteria`. ActivityTypeCriteria by which to filter ActivityType being retrieved.

Returns

List of ActivityTypeData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityPropertyLabel"
AssociatedControlID="uxActivityProperty"
            CssClass="span-3 last" runat="server" Text="Activity Property :" />
        <asp:DropDownList ID="uxActivityProperty" runat="server">

            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>ObjectType</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-3 last"
            runat="server" Text="*ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="20" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
```

```
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxActivityList" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Name
                    </th>
                    <th>
                        Object Type
                    </th>
                    <th>
                        SupportsComments
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectType")%>
            </td>
            <td class="devsite-method">
                <%# Eval("SupportsComments")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            string Property = uxActivityProperty.SelectedItem.Text;
            ActivityTypeManager activityTypeManager = new ActivityTypeManager();
            ActivityTypeCriteria criteria = new ActivityTypeCriteria();
            criteria.OrderByField = ActivityTypeProperty.Id;
            criteria.OrderByDirection = EkEnumeration.OrderByDirection.Ascending;
            if (Property == "Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(ActivityTypeProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (Property == "ObjectType")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(ActivityTypeProperty.ObjectType,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                criteria.AddFilter(ActivityTypeProperty.Name,
CriteriaFilterOperator.Contains, uxObjectValue.Text);
            }
            //Get Activity Type List for given Filter
            List<ActivityTypeData> activityTypeList = activityTypeManager.GetList
(criteria);

            uxActivityList.Visible = true;
            uxActivityList.DataSource = activityTypeList;
            uxActivityList.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Update

```
Update(Ektron.Cms.ActivityTypeData)
```

Updates an existing ActivityType item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity Type ID
- Name
- Enable Comments

Remarks

Validate the ActivityType item with `GetItem()` before you update the properties. Then, call `Update` with your modified ActivityTypeData object.

NOTE: You cannot change all properties after the initial Add event, such as the ActivityTypeData.Type.

Parameters

- `ActivityTypeData`–ActivityTypeData object to save.

Returns

Returns the custom ActivityTypeData object updated.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityTypeIDLabel"
AssociatedControlID="uxActivityTypeID"
            CssClass="span-4 last" runat="server" Text="*Activity TypeID :" />
        <ektronUI:TextField ID="uxActivityTypeID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-4
last"
            runat="server" Text="Name :" />
        <ektronUI:TextField ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxSupportsCommentsLabel"
AssociatedControlID="uxSupportsComments"
            CssClass="span-5 last" runat="server" Text="Enable Comments:" />
            <asp:DropDownList ID="uxSupportsComments" runat="server">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ActivityTypeManager activityTypeManager = new ActivityTypeManager();
            long activityTypeID = long.Parse(uxActivityTypeID.Text);
            //Check whether Entered activityTypeID valid or Not
            ActivityTypeData activityTypeData = activityTypeManager.GetItem
(activityTypeID);
            if (activityTypeData != null)
            {
                activityTypeData.Name = uxName.Text != string.Empty ? uxName.Text :
activityTypeData.Name;
                if (uxSupportsComments.SelectedValue.ToString() == "True")
                    activityTypeData.SupportsComments = true;
                else activityTypeData.SupportsComments = false;

                //Update Activity Type with given activityTypeDetails.
                activityTypeManager.Update(activityTypeData);
                MessageUtilities.UpdateMessage(uxMessage, "Activity Type with Id " +
activityTypeID + " has been Updated with given details.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,"Please Enter Valid Activity
```

```
Type ID", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Data Classes

ActivityTypeCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Activity
```

Constructors

- `ActivityTypeCriteria()`

  ```
  public class ActivityTypeCriteria :
  Ektron.Cms.Common.Criteria<ActivityTypeProperty>
  ```

- `ActivityTypeCriteria(Ektron.Cms.Activity.ActivityTypeProperty, EkEnumeration.OrderByDirection)`

  ```
  public ActivityTypeCriteria(Ektron.Cms.Activity.ActivityTypeProperty orderByField,
    EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `ActivityTypeProperty` are:

  - ActivityId
  - DisplayInPreferences
  - Id
  - IsPublishByDefault
  - Name
  - ObjectType
  - Scope
  - SupportsComments

ActivityTypeData

Namespace

```
Ektron.Cms.Activity
```

Properties

- `ActionType`. Gets or sets the action associated with the activity.

```
public EkEnumeration.ActivityActionType ActionType { set; get; }
```

- `DisplayInPreferences`. Gets or sets the preference display flag. If true, users can define notification preferences for the activity.

```
public bool DisplayInPreferences { set; get; }
```

- `IsPublishByDefault`

```
public bool IsPublishByDefault { set; get; }
```

- `Name`. Gets or sets the name of the ActivityType.

```
public string Name { set; get; }
```

- `ObjectType`. Gets or sets the CMS object type associated with activity. This may be null for custom activities.

```
public EkEnumeration.ActivityObjectType ObjectType { set; get; }
```

- `Scope`. Gets or sets the scope of the activity. For example, it may originate from a community group.

```
public EkEnumeration.ActivityScope Scope { set; get; }
```

- `SupportsComments`. Gets or sets the flag indicating if this ActivityType supports commenting.

```
public bool SupportsComments { set; get; }
```

Calendar

8.50 and higher

The Calendar manager category manages schedules and events and has the following classes:

- *WebCalendarManager* on the facing page. Manages calendars.
- *WebEventManager* on page 259. Manages schedules that appear on user calendars.

WebCalendarManager

8.50 and higher

The WebCalendarManager class manages calendars.

Namespace

```
Ektron.Cms.Framework.Calendar.WebCalendarManager
```

Constructors

- `WebCalendarManager()`
- `WebCalendarManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `CalendarService`. Calendar Service
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 242
- *GetCalendarFeed* on page 244
- *GetItem* on page 246
- *GetList* on page 248
- *GetPublicCalendar* on page 251 (public)
- *GetPublicCalendar* on page 253 (owner)
- *Update* on page 255

Add

```
Add(Ektron.Cms.Common.Calendar.WebCalendarData)
```

Adds a new Web Calendar to the CMS.

This method takes one argument for the WebCalendarData class. When you add the Web Calendar, the method returns a new WebCalendar ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Parent Folder ID
- * Web Calendar Name
- Description

Parameters

- `calendar`. WebCalendarData object to add.

Returns

Returns the WebCalendarData object added.

Remarks

Validate the parent folder using `GetItem()` before you add the Web calendar.

.aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxWebcalendarNameLabel"
      AssociatedControlID="uxWebcalendarName"
      CssClass="span-4 last" runat="server"
      Text="* Webcalendar Name :" />
    <ektronUI:TextField ID="uxWebcalendarName"
      CssClass="span-4" runat="server"
      ValidationGroup="RegisterValidationGroup" />
  </li>
  <li class="clearfix">
    <ektronUI:Label ID="uxParentFolderIDLabel"
      AssociatedControlID="uxParentFolderID"
      CssClass="span-4 last" runat="server"
      Text="Parent FolderID :" />
    <ektronUI:TextField ID="uxParentFolderID"
      CssClass="span-4" runat="server"
      ValidationGroup="RegisterValidationGroup" Text="0" />
  </li>
  <li class="clearfix">
    <ektronUI:Label ID="uxWebcalendarDescriptionLabel"
```

```
      AssociatedControlID="uxWebcalendarDescription"
      CssClass="span-4 last" runat="server" Text="Description :" />
    <asp:TextBox TextMode="MultiLine" Rows="3"
      ID="uxWebcalendarDescription" CssClass="span-4"
      runat="server" ValidationGroup="RegisterValidationGroup" />
  </li>
  <li class="clearfix" style="color: red;">
    <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9
      last" runat="server" Text="Status:" />
  </li>
  <ektronUI:Button ID="uxSubmit" runat="server"
     OnClick="uxSubmit_Click" Text="Add"></ektronUI:Button>
  <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4"
     runat="server" Text="* - Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
 {
   try
   {
     if (!string.IsNullOrEmpty(uxWebcalendarName.Text))
     {
       long folderID = Convert.ToInt64(uxParentFolderID.Text);

       FolderManager folderManager = new FolderManager();
       FolderData folderData = folderManager.GetItem(folderID);
       //Check whether entered folderID is Valid or NOT.
       if (folderData != null)
       {
         WebCalendarManager WebcalendarManager = new WebCalendarManager();

         //Set the WebCalendar Details
         WebCalendarData WebcalendarData = new WebCalendarData()
           {
             Name = uxWebcalendarName.Text,
             ParentId = folderID,
             Description = uxWebcalendarDescription.Text,
           };

          //Add a webcalender with given details.
```

```
          WebcalendarManager.Add(WebcalendarData);
          MessageUtilities.UpdateMessage(uxMessage,
            "WebCalendar Saved with Id " + WebcalendarData.Id,
            Message.DisplayModes.Success);
            }
              else
            {
              uxStatus.Visible = true;
              MessageUtilities.UpdateMessage(uxStatus,
                "Please Enter Valid Parent Folder ID.",
                Message.DisplayModes.Error);
              return;
             }
        }
          else
          {
            uxStatus.Visible = true;
            MessageUtilities.UpdateMessage(uxStatus,
            "Please Enter WebcalendarName and try again.",
            Message.DisplayModes.Error);
            return;
          }
          uxPageMultiView.SetActiveView(uxViewMessage);
      }
      catch (Exception ex)
      {
        MessageUtilities.UpdateMessage(uxMessage,
          ex.Message, Message.DisplayModes.Error);
        uxPageMultiView.SetActiveView(uxViewMessage);
      }
  }
```

Delete

```
Delete(System.Int64)
```

Deletes a Web Calendar from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID

Parameters

- `id`. The ID of the calendar to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxWebcalendarIDLabel"
       AssociatedControlID="uxWebcalendarID" CssClass="span-4 last"
       runat="server" Text="WebcalendarID :" />
    <ektronUI:TextField ID="uxWebcalendarID" CssClass="span-4"
       runat="server" ValidationGroup="RegisterValidationGroup" Text="0" />
  </li>
  <ektronUI:Button ID="uxSubmit" runat="server"
    OnClick="uxSubmit_Click" Text="Delete"></ektronUI:Button>
  <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4"
     runat="server" Text="* - Required" />
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
  {
    try
    {
      WebCalendarManager WebcalendarManager = new WebCalendarManager();
      long WebcalenderID = Convert.ToInt64(uxWebcalendarID.Text);

      if (WebcalenderID > 0)
      {
        WebCalendarData WebcalendarData =
          WebcalendarManager.GetItem(WebcalenderID);
        if (WebcalendarData != null)
        {
          //Delete a webcalender
          WebcalendarManager.Delete(WebcalenderID);
          MessageUtilities.UpdateMessage(uxMessage,
           "WebCalendar with ID: " + WebcalenderID +
           " has been Deleted", Message.DisplayModes.Success);
        }
        else
        {
          MessageUtilities.UpdateMessage(uxMessage,
           "Please Enter a Valid Webcalender ID",
           Message.DisplayModes.Error);
        }
```

```
      }
      else
      {
        MessageUtilities.UpdateMessage(uxMessage,
          "Please Enter a Valid Webcalender ID",
          Message.DisplayModes.Error);
      }
      uxPageMultiView.SetActiveView(uxViewMessage);
    }
    catch (Exception ex)
    {
      MessageUtilities.UpdateMessage(uxMessage,
        ex.Message, Message.DisplayModes.Error);
      uxPageMultiView.SetActiveView(uxViewMessage);
    }
  }
}
```

GetCalendarFeed

```
GetCalendarFeed(System.Int64)
```

Retrieves the Calendar representation of the supplied calendar ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID

Parameters

- `calendarId`. ID of calendar to retrieve ICalendar feed for.

.aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
     <ektronUI:Label ID="uxWebcalendarIDLabel"
        AssociatedControlID="uxWebcalendarID" CssClass="span-4 last"
        runat="server" Text="* Webcalendar ID :" />
     <ektronUI:TextField ID="uxWebcalendarID" CssClass="span-4"
        runat="server" ValidationGroup="RegisterValidationGroup" Text="88" />
  </li>
    <ektronUI:Button ID="uxSubmit" runat="server"
      OnClick="uxSubmit_Click" Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel"
      CssClass="span-4" runat="server" Text="* - Required" />
</ol>
```

```
<ol class="formFields">
  <li class="clearfix">
     <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
  </li>
  <li class="clearfix">
     <asp:Literal ID="uxFeed" runat="server"></asp:Literal>
  </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
  {
    try
    {
      WebCalendarManager WebcalendarManager = new WebCalendarManager();
      long WebcalenderID = Convert.ToInt64(uxWebcalendarID.Text);

      if (WebcalenderID > 0)
      {
        //Returns the ICalendar representation of the supplied calendar Id.
        String feed = WebcalendarManager.GetICalendarFeed(WebcalenderID);
        MessageUtilities.UpdateMessage(uxMessage,
          "ICalendar representation of the supplied calendar Id: "
          + WebcalenderID + " is below", Message.DisplayModes.Success);
        uxFeed.Text = feed;
       }
       else
       {
         MessageUtilities.UpdateMessage(uxMessage,
          "Please Enter a Valid Webcalender ID.",
            Message.DisplayModes.Error);
       }
       uxPageMultiView.SetActiveView(uxViewMessage);
     }
     catch (Exception ex)
     {
       MessageUtilities.UpdateMessage(uxMessage,
         ex.Message, Message.DisplayModes.Error);
       uxPageMultiView.SetActiveView(uxViewMessage);
     }
  }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific Web Calendar object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID

Parameters

- `calendarId`. The ID of the calendar to retrieve.

Returns

Retrieved WebCalendarData object.

.aspx code snippet

```
<ol class="formFields">
 <li class="clearfix">
   <ektronUI:Label ID="uxWebcalendarIDLabel"
     AssociatedControlID="uxWebcalendarID" CssClass="span-4 last"
     runat="server" Text="* Webcalendar ID :" />
   <ektronUI:TextField ID="uxWebcalendarID" CssClass="span-4"
     runat="server" ValidationGroup="RegisterValidationGroup" Text="88" />
 </li>
 <li class="clearfix" style="color: red;">
    <ektronUI:Label ID="uxStatus" Visible="false"
      CssClass="span-9 last" runat="server" Text="Status:" />
 </li>
 <ektronUI:Button ID="uxSubmit" runat="server"
    OnClick="uxSubmit_Click" Text="Add"></ektronUI:Button>
 <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4"
    runat="server" Text="* - Required" />
</ol>

<ol class="formFields">
  <li class="clearfix">
  <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
  </li>
  <li class="clearfix">
  <asp:Literal ID="uxWebcalendarName" runat="server"></asp:Literal>
  </li>
  <li class="clearfix">
  <asp:Literal ID="uxWebcalendarDescription" runat="server"></asp:Literal>
```

```
  </li>
  <li class="clearfix">
  <asp:Literal ID="uxWebcalendarParentID" runat="server"></asp:Literal>
  </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
  {
    try
    {
      WebCalendarManager WebcalendarManager = new WebCalendarManager();
      long WebcalenderID = Convert.ToInt64(uxWebcalendarID.Text);

      if (WebcalenderID > 0)
      {
        //Get a webcalender
        WebCalendarData WebcalendarData
          = WebcalendarManager.GetItem(WebcalenderID);
        if (WebcalendarData != null)
        {
          MessageUtilities.UpdateMessage(uxMessage,
            "WebCalendar Details for ID: " + WebcalenderID
            + " are below", Message.DisplayModes.Success);
          uxWebcalendarName.Text = "WebCalendar Name : "
            + WebcalendarData.Name;
          uxWebcalendarDescription.Text = "Description  : "
            + WebcalendarData.Description;
          uxWebcalendarParentID.Text = "Parent Folder ID : "
            + WebcalendarData.ParentId;
        }
        else
        {
          uxStatus.Visible = true;
          MessageUtilities.UpdateMessage(uxStatus,
            "Please Enter a Valid Webcalender ID",
            Message.DisplayModes.Error);
          return;
         }
      }
      else
      {
        uxStatus.Visible = true;
```

```
        MessageUtilities.UpdateMessage(uxStatus,
          "Please Enter a Valid Webcalender ID",
          Message.DisplayModes.Error);
        return;
      }
           uxPageMultiView.SetActiveView(uxViewMessage);
      }
      catch (Exception ex)
     {
       MessageUtilities.UpdateMessage(uxMessage, ex.Message,
         Message.DisplayModes.Error);
       uxPageMultiView.SetActiveView(uxViewMessage);
     }
  }
```

GetList

```
GetList(Ektron.Cms.Common.Calendar.WebEventCriteria)
```

Retrieves lists of objects through the GetList(WebEventCriteria) method.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Calendar Property
- * Object Value

Parameters

- `criteria`. Criteria used to retrieve calendars.

Returns

List with retrieved WebCalendarData items.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxCalendarPropertyLabel"
      AssociatedControlID="uxCalendarProperty"
      CssClass="span-4 last" runat="server" Text="*CalendarProperty :" />
    <asp:DropDownList ID="uxCalendarProperty" runat="server">
      <asp:ListItem>Id</asp:ListItem>
      <asp:ListItem>Title</asp:ListItem>
      <asp:ListItem>Description</asp:ListItem>
    </asp:DropDownList>
```

```
  </li>
  <li class="clearfix">
    <ektronUI:Label ID="uxObjectValueLabel"
      AssociatedControlID="uxObjectValue" CssClass="span-4 last"
      runat="server" Text="*ObjectValue :" />
    <ektronUI:TextField ID="uxObjectValue"
      CssClass="span-6" runat="server"
      ValidationGroup="RegisterValidationGroup" Text="88" />
  </li>
  <ektronUI:Button ID="uxSubmit" runat="server"
    OnClick="uxSubmit_Click" Text="GetList"></ektronUI:Button>
  <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4"
    runat="server" Text="* - Required" />
</ol>

<ol class="formFields">
  <li class="clearfix">
    <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
  </li>
  <li class="clearfix"></li>
  <asp:ListView ID="uxCalendarlistView" runat="server"
    ItemPlaceholderID="aspItemPlaceholder" Visible="false">
  <EmptyDataTemplate>
    Criteria did not match any records.</EmptyDataTemplate>
  <LayoutTemplate>
    <table class="devsite-api-method">
      <thead>
        <tr>
          <th>Id</th>
          <th>Name</th>
          <th>Description</th>
          <th>ParentId</th>
        </tr>
      </thead>
      <tbody>
        <asp:PlaceHolder ID="aspItemPlaceholder" runat="server">
        </asp:PlaceHolder>
      </tbody>
    </table>
  </LayoutTemplate>
  <ItemTemplate>
    <tr>
      <td class="devsite-method"><%# Eval("Id")%></td>
      <td class="devsite-method"><%# Eval("Name")%></td>
      <td class="devsite-method"><%# Eval("Description")%></td>
      <td class="devsite-method"><%# Eval("ParentId")%></td>
    </tr>
  </ItemTemplate>
  </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
  {
    try
    {
      WebCalendarManager WebcalendarManager = new WebCalendarManager();
      object Objectvalue;
      string Property = uxCalendarProperty.SelectedItem.Text;

      WebCalendarCriteria criteria = new WebCalendarCriteria();
      criteria.OrderByField = WebCalendarProperty.Id;
      criteria.OrderByDirection = EkEnumeration.OrderByDirection.Ascending;

      if (Property == "Id")
      {
        Objectvalue = Convert.ToInt64(uxObjectValue.Text);
        criteria.AddFilter(WebCalendarProperty.Id,
          CriteriaFilterOperator.EqualTo, Objectvalue);
      }
      else if (Property == "Title")
      {
        Objectvalue = uxObjectValue.Text;
        criteria.AddFilter(WebCalendarProperty.FolderName,
          CriteriaFilterOperator.Contains, Objectvalue);
      }
      else
      {
        Objectvalue = uxObjectValue.Text;
        criteria.AddFilter(WebCalendarProperty.Description,
          CriteriaFilterOperator.Contains, Objectvalue);
      }
      List<WebCalendarData> webCalendarList
        = WebcalendarManager.GetList(criteria);

      //Display the Web Calender List
      uxCalendarlistView.Visible = true;
      uxCalendarlistView.DataSource = webCalendarList;
      uxCalendarlistView.DataBind();

      uxPageMultiView.SetActiveView(uxViewMessage);
    }
    catch (Exception ex)
    {
      MessageUtilities.UpdateMessage(uxMessage,
      ex.Message, Message.DisplayModes.Error);
      uxPageMultiView.SetActiveView(uxViewMessage);
```

```
    }
  }
```

GetPublicCalendar

```
GetPublicCalendar(System.Int64)
```

Retrieves a calendar with only public fields populated.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Webcalendar ID (User ID or Group ID)

Parameters

- `calendarId`

Returns

WebCalendarData with only public fields populated.

.aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxWebcalendarIDLabel"
      AssociatedControlID="uxWebcalendarID" CssClass="span-4 last"
      runat="server" Text="* Webcalendar ID :" />
    <ektronUI:TextField ID="uxWebcalendarID" CssClass="span-4"
      runat="server" ValidationGroup="RegisterValidationGroup" Text="88" />
  </li>
  <ektronUI:Button ID="uxSubmit" runat="server"
    OnClick="uxSubmit_Click" Text="Add"></ektronUI:Button>
  <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4"
    runat="server" Text="* - Required" />
</ol>

<ol class="formFields">
  <li class="clearfix">
  <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
  </li>
  <li class="clearfix">
  <asp:Literal ID="uxWebcalendarName" runat="server"></asp:Literal>
  </li>
  <li class="clearfix">
  <asp:Literal ID="uxWebcalendarDescription" runat="server"></asp:Literal>
  </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
  {
    try
    {
      WebCalendarManager WebcalendarManager = new WebCalendarManager();
      long WebcalenderID = Convert.ToInt64(uxWebcalendarID.Text);

      if (WebcalenderID > 0)
      {
        //Get a Public webcalender
        WebCalendarData WebcalendarData
          = WebcalendarManager.GetPublicCalendar(WebcalenderID);
        if (WebcalendarData != null)
        {
          MessageUtilities.UpdateMessage(uxMessage,
            "WebCalendar Details for ID: " + WebcalenderID
            + " are below", Message.DisplayModes.Success);
          uxWebcalendarName.Text = "WebCalendar Name : "
            + WebcalendarData.Name;
          uxWebcalendarDescription.Text = "Description  : "
            + WebcalendarData.Description;
        }
        else
        {
          MessageUtilities.UpdateMessage(uxMessage,
            "Please Enter a Valid Webcalender ID",
            Message.DisplayModes.Error);
        }
      }
      else
      {
         MessageUtilities.UpdateMessage(uxMessage,
           "Please Enter a Valid Webcalender ID",
           Message.DisplayModes.Error);
      }
      uxPageMultiView.SetActiveView(uxViewMessage);
    }
    catch (Exception ex)
    {
      MessageUtilities.UpdateMessage(uxMessage, ex.Message,
        Message.DisplayModes.Error);
      uxPageMultiView.SetActiveView(uxViewMessage);
```

```
    }
  }
```

GetPublicCalendar

```
GetPublicCalendar(Ektron.Cms.Common.EkEnumeration.WorkSpace,System.Int64)
```

Get public calendar by owner type and owner ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Work Space
- * Owner ID (User ID or Group ID)

Parameters

- `ownerId`
- `ownerType`. Owner type either User or Group.

.aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxWorkSpaceLabel" AssociatedControlID="uxWorkSpace"
      CssClass="span-4 last" runat="server" Text="*WorkSpace :" />
    <asp:DropDownList ID="uxWorkSpace" runat="server">
      <asp:ListItem>UserId</asp:ListItem>
      <asp:ListItem>GroupID</asp:ListItem>
    </asp:DropDownList>
  </li>
  <div class="ektronTopSpace">
  </div>
  <li class="clearfix">
    <ektronUI:Label ID="uxOwnerIDLabel" AssociatedControlID="uxOwnerID"
      CssClass="span-4 last"
      runat="server" Text="*Owner ID (UserId/ GroupId) :" />
    <ektronUI:TextField ID="uxOwnerID" CssClass="span-4"
      runat="server" ValidationGroup="RegisterValidationGroup" Text="0" />
  </li>
  <div class="ektronTopSpace">
  </div>
  <li class="clearfix">
    <ektronUI:Button ID="uxSubmit" runat="server"
      OnClick="uxSubmit_Click" Text="GetCalendar"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4"
      runat="server" Text="* - Required" />
  </li>
```

```
</ol>

<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Message ID="uxMessage" DisplayMode="Information"
      Visible="false" runat="server" />
  </li>
  <li class="clearfix">
    <asp:Literal ID="uxWebcalendarName" runat="server"></asp:Literal>
  </li>
  <li class="clearfix">
    <asp:Literal ID="uxWebcalendarDescription" runat="server"></asp:Literal>
  </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
  {
    try
    {
      WebCalendarManager WebcalendarManager = new WebCalendarManager();
      long ownerID = Convert.ToInt64(uxOwnerID.Text);
      WebCalendarData WebcalendarData = new WebCalendarData();
      if (ownerID > 0)
      {
        if (uxWorkSpace.SelectedValue == "UserId")
        {
          //Get a Public webcalender
          WebcalendarData =
            WebcalendarManager.GetPublicCalendar(EkEnumeration.WorkSpace.User,
            ownerID);
        }
        else
        {
          //Get a Public webcalender
          WebcalendarData =
            WebcalendarManager.GetPublicCalendar(EkEnumeration.WorkSpace.Group,
            ownerID);
        }
        if (WebcalendarData != null)
        {
          MessageUtilities.UpdateMessage(uxMessage,
            "WebCalendar Details for ID: " + WebcalendarData.Id
```

```
            + " are below", Message.DisplayModes.Success);
        uxWebcalendarName.Text = "WebCalendar Name : "
            + WebcalendarData.Name;
        uxWebcalendarDescription.Text = "Description  : "
            + WebcalendarData.Description;
      }
      else
      {
        MessageUtilities.UpdateMessage(uxMessage,
          "Please Enter a Valid Owner ID", Message.DisplayModes.Error);
      }
    }
    else
    {
      MessageUtilities.UpdateMessage(uxMessage,
        "Please Enter a Valid Owner ID", Message.DisplayModes.Error);
    }
    uxPageMultiView.SetActiveView(uxViewMessage);
  }
  catch (Exception ex)
  {
    MessageUtilities.UpdateMessage(uxMessage, ex.Message,
      Message.DisplayModes.Error);
    uxPageMultiView.SetActiveView(uxViewMessage);
  }
}
```

Update

```
Update(Ektron.Cms.WebCalendarData)
```

Updates an existing WebCalendar item in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID
- Web Calendar Name
- Description

Parameters

- `calendar`. WebCalendarData object to update.

Returns

Returns the WebCalendarData object updated.

Remarks

Validate the WebCalendar item with `GetItem()` before you update the properties. Then, call `Update` with your modified WebCalendarData object.

NOTE: You cannot change all properties after the initial Add event, such as the WebCalendarData.Type.

.aspx code snippet

```
<ol class="formFields">
  <li class="clearfix">
    <ektronUI:Label ID="uxWebcalendarIDLabel"
      AssociatedControlID="uxWebcalendarID" CssClass="span-4 last"
      runat="server" Text="WebcalendarID :" />
    <ektronUI:TextField ID="uxWebcalendarID" CssClass="span-4"
      runat="server" ValidationGroup="RegisterValidationGroup" Text="0" />
  </li>
  <li class="clearfix">
    <ektronUI:Label ID="uxWebcalendarNameLabel"
      AssociatedControlID="uxWebcalendarName" CssClass="span-4 last"
      runat="server" Text="WebcalendarName :" />
    <ektronUI:TextField ID="uxWebcalendarName" CssClass="span-4"
      runat="server" ValidationGroup="RegisterValidationGroup" />
  </li>
  <li class="clearfix">
    <ektronUI:Label ID="uxWebcalendarDescriptionLabel"
      AssociatedControlID="uxWebcalendarDescription"
      CssClass="span-4 last" runat="server" Text="Description :" />
    <ektronUI:TextField ID="uxWebcalendarDescription"
      CssClass="span-4" runat="server"
      ValidationGroup="RegisterValidationGroup" />
  </li>
  <ektronUI:Button ID="uxSubmit" runat="server"
    OnClick="uxSubmit_Click" Text="Update"></ektronUI:Button>
  <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4"
    runat="server" Text="* - Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void Page_Init(object sender, EventArgs e)
  {
    ApiTitle = "Ektron.Cms.Framework.Calendar.WebCalendarManager";
    SelectedAccordion = "uxTabCalendar";
    ComponentName = "Ektron.Cms.Framework.Calendar.WebCalendarManager";
    ComponentUrl = "Framework/Calendar/WebCalendarManager/";
    DocumentationName = "Ektron.Cms.Framework.Calendar.WebCalendarManager.Update
(Ektron.Cms.Common.Calendar.WebCalendarData)";
    this.IsDemoSafe = false;
   // CBWebCalendarManager.DefaultContentID = ContentID;
  }
  protected void uxSubmit_Click(object sender, EventArgs e)
  {
    try
    {
      WebCalendarManager WebcalendarManager = new WebCalendarManager();

      //Set the WebCalendar Details
      WebCalendarData WebcalendarData =
        WebcalendarManager.GetItem(Convert.ToInt64(uxWebcalendarID.Text));
      if (WebcalendarData != null)
      {
        WebcalendarData.Name = uxWebcalendarName.Text != string.Empty
          ? uxWebcalendarName.Text : WebcalendarData.Name;
        WebcalendarData.Description = uxWebcalendarDescription.Text
          != string.Empty ? uxWebcalendarDescription.Text
          : WebcalendarData.Description;

        //Update webcalender with given details.
        WebcalendarManager.Update(WebcalendarData);
        MessageUtilities.UpdateMessage(uxMessage,
          "WebCalendar with ID: " + WebcalendarData.Id
            + " has been updated", Message.DisplayModes.Success);
       }
       else
       {
         MessageUtilities.UpdateMessage(uxMessage,
           "Please Enter Valid Calendar ID", Message.DisplayModes.Error);
       }
       uxPageMultiView.SetActiveView(uxViewMessage);
     }
     catch (Exception ex)
     {
       MessageUtilities.UpdateMessage(uxMessage, ex.Message,
         Message.DisplayModes.Error);
       uxPageMultiView.SetActiveView(uxViewMessage);
     }
  }
```

Data Classes

WebCalendarCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Common.Calendar
```

Constructors

- `WebCalendarCriteria()`

  ```
  public WebCalendarCriteria()
  ```

- `WebCalendarCriteria(Ektron.Cms.Common.WebCalendarProperty, EkEnumeration.OrderByDirection)`

  ```
  public WebCalendarCriteria(Ektron.Cms.Common.WebCalendarProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `WebCalendarProperty` are:

  - Description
  - FolderName
  - FolderPath
  - Id
  - ParentId

WebCalendarData

Namespace

```
Ektron.Cms.Common.Calendar
```

Properties

- `IsValid(ValidationResults)`. Validates the Calendar object and returns a list of errors.

  ```
  public bool IsValid(ValidationResults validationResult)
  ```

- `LoadFolderData(Ektron.Cms.FolderData)`

  ```
  public void LoadFolderData(Ektron.Cms.FolderData folderData)
  ```

- `WebCalendarData()`

  ```
  public WebCalendarData()
  ```

WebEventManager

8.50 and higher

The WebEventManager class manages schedules that appear on user calendars.

Namespace

```
Ektron.Cms.Framework.Calendar.WebEventManager
```

Constructors

- `WebEventManager()`
- `WebEventManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.
- `WebEventManagerService`. WebEvent manager service.

Methods

- *GetList (taxonomy)* on page 295
- *GetList (ID)* on page 292
- *GetList (date/time)* on page 288
- *GetNonVariantEventList (ID)* on page 304
- *GetNonVariantEventList (date/time)* on page 300
- *GetVariantEventList (ID)* on page 311
- *GetVariantEventList (date/time)* on page 307
- *Update* on page 314

Add

```
Add(Ektron.Cms.Common.Calendar.WebEventData)
```

Adds a new event to a Web Calendar.

This method takes one argument for the WebEventData class. When you add the Web Calendar, the method returns a new WebEvent ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Event Title
- * Web Calendar ID
- Location
- Description
- * Start Date Time
- * End Date Time

Parameters

- `eventData`. WebEventData object to add.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebEventTitleLabel" AssociatedControlID="uxWebEventTitle"
CssClass="span-4 last"
            runat="server" Text="* WebEvent Title :" />
        <ektronUI:TextField ID="uxWebEventTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxWebcalendarIDLabel" AssociatedControlID="uxWebcalendarID"
CssClass="span-4 last"
            runat="server" Text="* WebcalendarID :" />
        <ektronUI:TextField ID="uxWebcalendarID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLocationLabel" AssociatedControlID="uxLocation"
CssClass="span-4 last"
            runat="server" Text="Location :" />
        <ektronUI:TextField ID="uxLocation" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxWebEventDescriptionLabel"
AssociatedControlID="uxWebEventDescription"
            CssClass="span-4 last" runat="server" Text="Description :" />
        <asp:TextBox TextMode="MultiLine" Rows="3" ID="uxWebEventDescription"
CssClass="span-4"
            runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-4 last"
            runat="server" Text="* Start DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>

        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEndDateLabel" AssociatedControlID="uxEndDate"
CssClass="span-4 last"
            runat="server" Text="* End DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxEndDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
```

```
        <asp:DropDownList ID="uxEndTime" runat="server">
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
            <asp:ListItem>12</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEndPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            if (!string.IsNullOrEmpty(uxWebEventTitle.Text))
            {
                WebEventManager webeventManager = new WebEventManager();
                String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text +
":00:00 " + uxStartPeriod.SelectedItem.Text;
                String et = uxEndDate.Text + " " + uxEndTime.SelectedItem.Text + ":00:00
" + uxEndPeriod.SelectedItem.Text;
                DateTime startdatetime = DateTime.Parse(st);
                DateTime endDatetime = DateTime.Parse(et);

                //Set the WebEvent Details
```

```
                WebEventData WebeventData = new WebEventData()
                {

                    Title = uxWebEventTitle.Text,
                    FolderId = Convert.ToInt64(uxWebcalendarID.Text),
                    Location = uxLocation.Text,
                    Description = uxWebEventDescription.Text,
                    EventStart = startdatetime,
                    EventEnd = endDatetime,

                };

                //Add a webEvent with given details.
                webeventManager.Add(WebeventData);
                MessageUtilities.UpdateMessage(uxMessage, "WebEvent Saved with Id " +
WebeventData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter WebEvent title
and try again", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

CancelOccurrence

```
CancelOccurrence(System.Int64, System.DateTime)
```

Cancels an occurrence of a recurring event.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Recurring Web Event ID
- * Event Occurrence Date

Parameters

- `eventId`. ID of the recurring event for which to cancel an occurrence.
- `occurrenceStartUtc`. The event occurrence date to cancel.

Returns

ID of the event variance created for the cancellation.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebEventIDLabel" AssociatedControlID="uxWebEventID"
CssClass="span-4 last"
            runat="server" Text="*Recurring WebEventID :" />
        <ektronUI:TextField ID="uxWebEventID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-4 last"
            runat="server" Text="*EventOccurance Date :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            WebEventManager webeventManager = new WebEventManager();
            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;

            DateTime startdatetime = DateTime.Parse(st).ToUniversalTime();

            long eventid = Convert.ToInt64(uxWebEventID.Text);

            //Check the WebEvent Details
            WebEventData WebeventData = webeventManager.GetItem(eventid);

            if (WebeventData != null)
            {
                //Cancels an occurence of a recurring event.
                long cancelID = webeventManager.CancelOccurrence(eventid,
startdatetime);
                MessageUtilities.UpdateMessage(uxMessage, "Canceled an occurence of a
supplied recurring Web Event ID : " + eventid, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Web
Calendar Event ID and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

CreateVariance

```
CreateVariance(System.Int64, System.DateTime)
```

Creates a WebEventData object representing variance event for the supplied recurring event ID. The WebEventData object is not yet saved and should be passed to `Add()` when ready for saving.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Recurring Web Event ID
- * Event Occurrence Date

Parameters

- `eventId`. ID of the recurring event for which to create a variance.
- `occurrenceStartUtc`. The event occurrence date for which to create a variance.

Returns

An unsaved WebEvent representing the event variance.

Remarks

You should pass the WebEventData to `Add()` when you are ready to save the object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebEventIDLabel" AssociatedControlID="uxWebEventID"
CssClass="span-4 last"
            runat="server" Text="*Recurring WebEventID :" />
        <ektronUI:TextField ID="uxWebEventID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="251" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-4 last"
            runat="server" Text="*EventOccurance Date :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
```

```
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            WebEventManager webeventManager = new WebEventManager();
            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;

            DateTime startdatetime = DateTime.Parse(st);

            long eventid = Convert.ToInt64(uxWebEventID.Text);

            //Check the WebEvent Details
            WebEventData webEventData = webeventManager.GetItem(eventid);

            if (webEventData != null)
            {
                //Creates a variance event for the supplied recurring event id.
                webEventData = webeventManager.CreateVariance(eventid, startdatetime);
```

```
                uxTitle.Text = webEventData.Title;
                uxDate.Text = webEventData.EventStart.ToShortDateString() + " " +
webEventData.EventStart.ToShortTimeString();
                MessageUtilities.UpdateMessage(uxMessage,  "Created a variance event for
the supplied recurring Web Event ID : " + eventid, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Web
Calendar Event ID and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }


    }
```

CreateVariance

```
CreateVariance(System.Int64, System.DateTime, System.DateTime)
```

Creates a WebEventData object representing variance event for the supplied recurring event id. The WebEventData object has not been saved yet and should be passed to `Add()` when ready for saving.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Recurring Web Event ID
- * Event Occurrence Date
- * Start Date Time

Parameters

- `eventId`. ID of the recurring event for which to create a variance.
- `occurrenceStartUtc`. The event occurrence date for which to create a variance.
- `newStartUtc`. The new occurrence start date; Coordinated Universal Time (UTC).

Returns

An unsaved WebEvent representing the event variance.

Remarks

You should pass the WebEventData to `Add()` when you are ready to save the object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebEventIDLabel" AssociatedControlID="uxWebEventID"
CssClass="span-4 last"
            runat="server" Text="*Recurring WebEventID :" />
        <ektronUI:TextField ID="uxWebEventID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="251" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEventOccuranceDateLabel"
AssociatedControlID="uxEventOccuranceDate"
            CssClass="span-4 last" runat="server" Text="*EventOccurance Date :" />
        <ektronUI:DatePicker Rows="3" ID="uxEventOccuranceDate" CssClass="span-4"
runat="server"
            Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxEventOccuranceTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEventOccurancePeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-4 last"
            runat="server" Text="*Start DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
```

```
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            WebEventManager webeventManager = new WebEventManager();
            String et = uxEventOccuranceDate.Text + " " +
uxEventOccuranceTime.SelectedItem.Text + ":00:00 " +
uxEventOccurancePeriod.SelectedItem.Text;
            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;

            DateTime EventOccuranceDatetime = DateTime.Parse(et);
            DateTime startdatetime = DateTime.Parse(st);

            long eventid = Convert.ToInt64(uxWebEventID.Text);

            //Check the WebEvent Details
            WebEventData webEventData = webeventManager.GetItem(eventid);
```

```
            if (webEventData != null)
            {
                //Creates a variance event for the supplied recurring event id.
                webEventData = webeventManager.CreateVariance(eventid,
EventOccuranceDatetime, startdatetime);
                uxTitle.Text = webEventData.Title;
                uxDate.Text = webEventData.EventStart.ToShortDateString() + " " +
webEventData.EventStart.ToShortTimeString();
                MessageUtilities.UpdateMessage(uxMessage, "Created a variance event for
the supplied recurring Web Event ID : " + eventid, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Web
Calendar Event ID and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }


    }
```

Delete

```
Delete(System.Int64)
```

Deletes a Web calendar event from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Event ID

Parameters

- `id`. ID of WebEvent to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebEventIDLabel" AssociatedControlID="uxWebEventID"
CssClass="span-4 last"
            runat="server" Text="* WebEventID :" />
        <ektronUI:TextField ID="uxWebEventID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            WebEventManager webeventManager = new WebEventManager();
            long eventid = Convert.ToInt64(uxWebEventID.Text);

            //Check the WebEvent Details
            WebEventData WebeventData = webeventManager.GetItem(eventid);
            if (WebeventData != null)
            {
                //Delete a webEvent with given ID.
                webeventManager.Delete(eventid);
                MessageUtilities.UpdateMessage(uxMessage, "Web Event ID : " +
WebeventData.Id + " has been Deleted from the Calendar.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid WebEvent
ID and Try again.", Message.DisplayModes.Error);
            }
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetEventOccurrenceList (date/time)

```
GetEventOccurrenceList(System.Int64,System.DateTime,System.DateTime,System.Boolean)
```

Returns a list of events for the calendar that occur during the supplied time period. This will return all event occurrences including individual occurrences for recurring events.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID
- * Start Date Time
- * End Date Time

Parameters

- `calendarId`. ID of the calendar to retrieve events for.
- `startUtc`. Start of search span.
- `endUtc`. End of search span.

Returns

Retrieved list of WebEventData objects.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebCalendarIDLabel" AssociatedControlID="uxWebCalendarID"
CssClass="span-4 last"
            runat="server" Text="*WebCalendar ID :" />
        <ektronUI:TextField ID="uxWebCalendarID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="88" />
    </li>
```

```
        <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-4 last"
            runat="server" Text="*Start DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEndDateLabel" AssociatedControlID="uxEndDate"
CssClass="span-4 last"
            runat="server" Text="*End DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxEndDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxEndTime" runat="server">
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
            <asp:ListItem>12</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEndPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
```

```
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix"></li>
    <asp:ListView ID="uxEventlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
        Visible="false">
        <EmptyDataTemplate>
            Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
                        <th>
                            WebEvent Id
                        </th>
                        <th>
                            Title
                        </th>
                        <th>
                            Location
                        </th>
                        <th>
                            Start Date
                        </th>
                        <th>
                            End Date
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("Id")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Title")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Location")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventStart")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventEnd")%>
                </td>
```

```
            </tr>
        </ItemTemplate>
    </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            WebEventManager webeventManager = new WebEventManager();
            long calendarId = Convert.ToInt64(uxWebCalendarID.Text);
            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;
            String et = uxEndDate.Text + " " + uxEndTime.SelectedItem.Text + ":00:00 " +
uxEndPeriod.SelectedItem.Text;
            DateTime startdatetime = DateTime.Parse(st);
            DateTime endDatetime = DateTime.Parse(et);
            //Check the WebEvent Details
            List<WebEventData> webeventList = webeventManager.GetEventOccurrenceList
(calendarId, startdatetime, endDatetime);
            if (webeventList != null)
            {
                //Display the Web Event Details
                uxEventlistView.Visible = true;
                uxEventlistView.DataSource = webeventList;
                uxEventlistView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Calendar
ID and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

GetEventOccurrenceList (event taxonomy)

8.60 and higher

```
GetEventOccurrenceList(System.Int64,System.DateTime,System.DateTime,
System.Collections.Generic.List)
```

Retrieves a list of events for the calendar that occur during the supplied time period. This will return all event occurrences including individual occurrences for recurring events.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID
- * Start Date Time
- * End Date Time
- Taxonomy Property
- * Object Value

Parameters

- `criteria`. Criteria used to restrict Web Event retrieval.

Returns

Retrieved list of WebEventData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
                    <li class="clearfix">
                        <ektronUI:Label ID="uxEventPropertyLabel"
AssociatedControlID="uxEventProperty" CssClass="span-3 last"
                            runat="server" Text="*EventTaxonomyProperty :" />
                        <asp:DropDownList ID="uxEventProperty" runat="server">
                            <asp:ListItem>Taxonomy Id</asp:ListItem>
                            <asp:ListItem>Id</asp:ListItem>
                        </asp:DropDownList>
                    </li>
                    <div class="ektronTopSpace"></div>
                    <li class="clearfix">
```

```
                        <ektronUI:Label ID="uxObjectValueLabel"
AssociatedControlID="uxObjectValue" CssClass="span-3 last"
                            runat="server" Text="*ObjectValue :" />
                        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup"
                            Text="189" />
                    </li>
                    <li class="clearfix">
                        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_
Click" Text="GetList"></ektronUI:Button>
                        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4"
runat="server" Text="* - Required" />
                    </li>
                </ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            var Property = uxEventProperty.SelectedItem.Text;

            WebEventManager webeventManager = new WebEventManager();
            EventTaxonomyCriteria criteria = new EventTaxonomyCriteria();
            criteria.OrderByField = EventProperty.Title;
            criteria.OrderByDirection = EkEnumeration.OrderByDirection.Descending;

            if (Property == "Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(EventProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(EventTaxonomyProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            //Check the WebEvent Details
```

```
            List<WebEventData> webeventList = webeventManager.GetEventOccurrenceList
(criteria);
            if (webeventList != null)
            {
                //Display the Web Event Details
                uxEventlistView.Visible = true;
                uxEventlistView.DataSource = webeventList;
                uxEventlistView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid  ID and
Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetEventOccurrenceList (taxonomy)

```
GetEventOccurrenceList
(System.Int64,System.DateTime,System.DateTime,System.Collections.Generic.List,System.Boo
lean)
```

Returns a list of events for the calendar that occur during the supplied time period. This will return all event occurrences including individual occurrences for recurring events.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID
- * Start Date Time
- * End Date Time
- Taxonomy Property
- * Object Value

Parameters

- `calendarId`. ID of the calendar to retrieve events for.
- `startUtc`. Start of search span.
- `endUtc`. End of search span.
- `TaxonomyIds`. List of the taxonomies assigned to the event.

Returns

Retrieved list of WebEventData objects.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="Label1" AssociatedControlID="uxWebCalendarID"
CssClass="span-4 last"
            runat="server" Text="*WebCalendar ID :" />
        <ektronUI:TextField ID="TextField1" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="88" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="Label2" AssociatedControlID="uxStartDate" CssClass="span-4
last"
            runat="server" Text="*Start DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="DatePicker1" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="DropDownList1" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="DropDownList2" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="Label3" AssociatedControlID="uxEndDate" CssClass="span-4
last"
            runat="server" Text="*End DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="DatePicker2" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
```

```
        <asp:DropDownList ID="DropDownList3" runat="server">
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
            <asp:ListItem>12</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="DropDownList4" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="Label4" AssociatedControlID="uxTaxonomyProperty"
CssClass="span-4 last"
            runat="server" Text="Taxonomy Property:" />
        <asp:DropDownList ID="DropDownList5" runat="server">
            <asp:ListItem>Taxonomy Id</asp:ListItem>
            <asp:ListItem>Taxonomy Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="Label5" AssociatedControlID="uxObjectValue" CssClass="span-4
last"
            runat="server" Text="* Object Value:" />
        <ektronUI:TextField ID="TextField2" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="189" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="Button1" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="Label6" CssClass="span-4" runat="server" Text="* - Required"
/>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
                        object Objectvalue;

                        TaxonomyCriteria criteria = new TaxonomyCriteria();
                        if (uxTaxonomyProperty.SelectedItem.Text == "Taxonomy Id")
                        {
                                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                                criteria.AddFilter(TaxonomyProperty.Id, CriteriaFilterOperator.EqualT
Objectvalue);
                        }
                        else
                        {
                                Objectvalue = uxObjectValue.Text;
                                criteria.AddFilter(TaxonomyProperty.Name, CriteriaFilterOperator.Cont
Objectvalue);
                        }

                        TaxonomyManager taxonomyManager = new TaxonomyManager();
                        List<TaxonomyData> taxonomyItemList = taxonomyManager.GetList(criteria);

            WebEventManager webeventManager = new WebEventManager();
            long calendarId = Convert.ToInt64(uxWebCalendarID.Text);
            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;
            String et = uxEndDate.Text + " " + uxEndTime.SelectedItem.Text + ":00:00 " +
uxEndPeriod.SelectedItem.Text;
            DateTime startdatetime = DateTime.Parse(st);
            DateTime endDatetime = DateTime.Parse(et);
            //Check the WebEvent Details
            List<WebEventData> webeventList = webeventManager.GetEventOccurrenceList
(calendarId, startdatetime, endDatetime, taxonomyItemList.ConvertAll(IdsOnly));
            if (webeventList != null)
            {
                //Display the Web Event Details
                uxEventlistView.Visible = true;
                uxEventlistView.DataSource = webeventList;
                uxEventlistView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Calendar
ID and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }

       private static long IdsOnly(TaxonomyData input)
       {
               return input.Id;
       }
```

GetEventOccurrenceList (Web event)

8.60 and higher

```
GetEventOccurrenceList(System.Int64,System.DateTime,System.DateTime)
```

Retrieves a list of events for the calendar that occur during the supplied time period. This will return all event occurrences including individual occurrences for recurring events.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID
- * Start Date Time
- * End Date Time

Parameters

- `criteria`. Criteria used to restrict WebEvent retrieval.

Returns

Retrieved list of WebEventData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEventPropertyLabel" AssociatedControlID="uxEventProperty"
CssClass="span-3 last"
            runat="server" Text="*EventProperty :" />
        <asp:DropDownList ID="uxEventProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Title</asp:ListItem>
            <asp:ListItem>Location</asp:ListItem>
        </asp:DropDownList>
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-3 last"
            runat="server" Text="*ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="251" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix"></li>
    <asp:ListView ID="uxEventlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
        Visible="false">
        <EmptyDataTemplate>
            Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
                        <th>
                            WebEvent Id
                        </th>
                        <th>
                            Title
                        </th>
                        <th>
                            Location
                        </th>
                        <th>
                            Start Date
                        </th>
                        <th>
                            End Date
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("Id")%>
```

```
                </td>
                <td class="devsite-method">
                    <%# Eval("Title")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Location")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventStart")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventEnd")%>
                </td>
            </tr>
        </ItemTemplate>
    </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            string Property = uxEventProperty.SelectedItem.Text;
            WebEventManager webeventManager = new WebEventManager();
            WebEventCriteria criteria = new WebEventCriteria();
            criteria.OrderByField = EventProperty.Title;
            criteria.OrderByDirection = EkEnumeration.OrderByDirection.Descending;

            if (Property == "Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(EventProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else if (Property == "Title")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(EventProperty.Title, CriteriaFilterOperator.Contains,
Objectvalue);
            }
            else
```

```
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(EventProperty.Location,
CriteriaFilterOperator.Contains, Objectvalue);
            }
            List<WebEventData> webEventList = webeventManager.GetEventOccurrenceList
(criteria);

            //Display the Web Event Details
            uxEventlistView.Visible = true;
            uxEventlistView.DataSource = webEventList;
            uxEventlistView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }


    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific WebEvent object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Event ID

Parameters

- `eventId`. ID of the event to retrieve.

Returns

Retrieved WebEventData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebEventIDLabel" AssociatedControlID="uxWebEventID"
```

```
CssClass="span-4 last"
            runat="server" Text="* WebEventID :" />
        <ektronUI:TextField ID="uxWebEventID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="251" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix"></li>
    <li class="clearfix">
        <asp:Literal ID="uxWebEventTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxWebEventDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxWebEventLocation" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxWebEventStartDate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxWebEventEndDate" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            WebEventManager webeventManager = new WebEventManager();
            long eventid = Convert.ToInt64(uxWebEventID.Text);
```

```
            //Check the WebEvent Details
            WebEventData WebeventData = webeventManager.GetItem(eventid);
            if (WebeventData != null)
            {


                //Display the Web Event Details
                MessageUtilities.UpdateMessage(uxMessage, "Web Event Details for ID : "
+ WebeventData.Id + " are below", Message.DisplayModes.Success);
                uxWebEventTitle.Text = "Web Event Title : " + WebeventData.Title;
                uxWebEventDescription.Text = "Description : " +
WebeventData.Description;
                uxWebEventLocation.Text = "Location : " + WebeventData.Location;
                uxWebEventStartDate.Text = "Start Date : " + WebeventData.EventStart;
                uxWebEventEndDate.Text = " End Date : " + WebeventData.EventEnd;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,"Please Enter Valid WebEvent ID
and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetList (date/time)

```
GetList(System.Int64,System.DateTime,System.DateTime)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID
- * Start Date Time
- * End Date Time

Parameters

- `calendarId`. ID of the calendar for which to retrieve events.
- `startUtc`. Start of time period to retrieve.
- `endUtc`. End of time period to retrieve.

Returns

Retrieved list of WebEventData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebCalendarIDLabel" AssociatedControlID="uxWebCalendarID"
CssClass="span-4 last"
            runat="server" Text="*WebCalendar ID :" />
        <ektronUI:TextField ID="uxWebCalendarID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="88" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-4 last"
            runat="server" Text="*Start DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEndDateLabel" AssociatedControlID="uxEndDate"
CssClass="span-4 last"
            runat="server" Text="*End DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxEndDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
```

```
        <asp:DropDownList ID="uxEndTime" runat="server">
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
            <asp:ListItem>12</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEndPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix"></li>
    <asp:ListView ID="uxEventlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
        Visible="false">
        <EmptyDataTemplate>
            Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
                        <th>
                            WebEvent Id
                        </th>
                        <th>
                            Title
                        </th>
                        <th>
                            Location
                        </th>
                        <th>
                            Start Date
                        </th>
                        <th>
                            End Date
                        </th>
                    </tr>
                </thead>
```

```
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("Id")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Title")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Location")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventStart")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventEnd")%>
                </td>
            </tr>
        </ItemTemplate>
    </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            WebEventManager webeventManager = new WebEventManager();
            long calendarId = Convert.ToInt64(uxWebCalendarID.Text);
            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;
            String et = uxEndDate.Text + " " + uxEndTime.SelectedItem.Text + ":00:00 " +
uxEndPeriod.SelectedItem.Text;
            DateTime startdatetime = DateTime.Parse(st);
            DateTime endDatetime = DateTime.Parse(et);
            //Check the WebEvent Details
```

```
            List<WebEventData> webeventList = webeventManager.GetList(calendarId,
startdatetime, endDatetime);
            if (webeventList != null)
            {
                //Display the Web Event Details
                uxEventlistView.Visible = true;
                uxEventlistView.DataSource = webeventList;
                uxEventlistView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Calendar
ID and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }


    }
```

GetList (ID)

```
GetList(System.Int64)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID

Parameters

- `calendarId`. ID of the calendar for which to retrieve events.

Returns

Retrieved list of WebEventData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebCalendarIDLabel" AssociatedControlID="uxWebCalendarID"
CssClass="span-4 last"
            runat="server" Text="*WebCalendar ID :" />
        <ektronUI:TextField ID="uxWebCalendarID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="88" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix"></li>
    <asp:ListView ID="uxEventlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
        Visible="false">
        <EmptyDataTemplate>
            Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
                        <th>
                            WebEvent Id
                        </th>
                        <th>
                            Title
                        </th>
                        <th>
                            Location
                        </th>
                        <th>
                            Start Date
                        </th>
                        <th>
                            End Date
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
        <ItemTemplate>
            <tr>
```

```
                <td class="devsite-method">
                    <%# Eval("Id")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Title")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Location")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventStart")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventEnd")%>
                </td>
            </tr>
        </ItemTemplate>
    </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            WebEventManager webeventManager = new WebEventManager();
            long calendarId = Convert.ToInt64(uxWebCalendarID.Text);

            //Check the WebEvent Details
            List<WebEventData> webeventList = webeventManager.GetList(calendarId);
            if (webeventList != null)
            {
                //Display the Web Event Details
                uxEventlistView.Visible = true;
                uxEventlistView.DataSource = webeventList;
                uxEventlistView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Calendar
ID and Try again.", Message.DisplayModes.Error);
            }
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }


    }
```

GetList (taxonomy)

8.60 and higher

```
GetList(Ektron.Cms.EventTaxonomyCriteria)
```

Returns a list of WebEvents based upon the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Event Property
- * Object Value

Parameters

- `criteria`. Criteria used to restrict WebEvent retrieval.

Returns

Retrieved list of WebEventData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
                    <li class="clearfix">
                        <ektronUI:Label ID="uxEventPropertyLabel"
AssociatedControlID="uxEventProperty" CssClass="span-3 last"
                            runat="server" Text="*EventTaxonomyProperty :" />
                        <asp:DropDownList ID="uxEventProperty" runat="server">
                            <asp:ListItem>Taxonomy Id</asp:ListItem>
                            <asp:ListItem>Id</asp:ListItem>
                        </asp:DropDownList>
```

```
                    </li>
                    <div class="ektronTopSpace"></div>
                    <li class="clearfix">
                        <ektronUI:Label ID="uxObjectValueLabel"
AssociatedControlID="uxObjectValue" CssClass="span-3 last"
                            runat="server" Text="*ObjectValue :" />
                        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup"
                            Text="189" />
                    </li>
                    <li class="clearfix">
                        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_
Click" Text="GetList"></ektronUI:Button>
                        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4"
runat="server" Text="* - Required" />
                    </li>
                </ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            var Property = uxEventProperty.SelectedItem.Text;

            WebEventManager webeventManager = new WebEventManager();
            EventTaxonomyCriteria criteria = new EventTaxonomyCriteria();
            criteria.OrderByField = EventProperty.Title;
            criteria.OrderByDirection = EkEnumeration.OrderByDirection.Descending;

            if (Property == "Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(EventProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(EventTaxonomyProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
```

```
            }

            //Check the WebEvent Details
            List<WebEventData> webeventList = webeventManager.GetList(criteria);
            if (webeventList != null)
            {
                //Display the Web Event Details
                uxEventlistView.Visible = true;
                uxEventlistView.DataSource = webeventList;
                uxEventlistView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid  ID and
Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (Web event)

```
GetList(Ektron.Cms.Common.Calendar.WebEventCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Event Property
- * Object Value

Parameters

- `criteria`. Criteria used to restrict WebEvent retrieval.

Returns

Retrieved list of WebEventData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEventPropertyLabel" AssociatedControlID="uxEventProperty"
CssClass="span-3 last"
            runat="server" Text="*EventProperty :" />
        <asp:DropDownList ID="uxEventProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Title</asp:ListItem>
            <asp:ListItem>Location</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-3 last"
            runat="server" Text="*ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="251" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix"></li>
    <asp:ListView ID="uxEventlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
        Visible="false">
        <EmptyDataTemplate>
            Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
                        <th>
                            WebEvent Id
                        </th>
                        <th>
                            Title
                        </th>
                        <th>
                            Location
                        </th>
                        <th>
                            Start Date
                        </th>
                        <th>
                            End Date
                        </th>
```

```
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("Id")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Title")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Location")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventStart")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventEnd")%>
                </td>
            </tr>
        </ItemTemplate>
    </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            string Property = uxEventProperty.SelectedItem.Text;
            WebEventManager webeventManager = new WebEventManager();
            WebEventCriteria criteria = new WebEventCriteria();
            criteria.OrderByField = EventProperty.Title;
            criteria.OrderByDirection = EkEnumeration.OrderByDirection.Descending;

            if (Property == "Id")
```

```
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(EventProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else if (Property == "Title")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(EventProperty.Title, CriteriaFilterOperator.Contains,
Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(EventProperty.Location,
CriteriaFilterOperator.Contains, Objectvalue);
            }
            List<WebEventData> webEventList = webeventManager.GetList(criteria);

            //Display the Web Event Details
            uxEventlistView.Visible = true;
            uxEventlistView.DataSource = webEventList;
            uxEventlistView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }


    }
```

GetNonVariantEventList (date/time)

```
GetVariantEventList(System.Int64,System.DateTime,System.DateTime)
```

Returns a list of original event data items for the calendar. For recurring events, this only returns the actual event items and not the occurrences or variants of recurring events.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID
- * Start Date Time
- * End Date Time

Parameters

- `calendarId`. ID of the calendar for which to retrieve events.
- `startUtc`. Start of time period to retrieve.
- `endUtc`. End of time period to retrieve.

Returns

Retrieved list of WebEventData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebCalendarIDLabel" AssociatedControlID="uxWebCalendarID"
CssClass="span-4 last"
            runat="server" Text="*WebCalendar ID :" />
        <ektronUI:TextField ID="uxWebCalendarID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="88" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-4 last"
            runat="server" Text="*Start DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEndDateLabel" AssociatedControlID="uxEndDate"
```

```
CssClass="span-4 last"
            runat="server" Text="*End DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxEndDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxEndTime" runat="server">
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
            <asp:ListItem>12</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEndPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix"></li>
    <asp:ListView ID="uxEventlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
        Visible="false">
        <EmptyDataTemplate>
            Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
                        <th>
                            WebEvent Id
                        </th>
                        <th>
                            Title
                        </th>
                        <th>
                            Location
                        </th>
                        <th>
                            Start Date
                        </th>
```

```
                        <th>
                            End Date
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("Id")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Title")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Location")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventStart")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventEnd")%>
                </td>
            </tr>
        </ItemTemplate>
    </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            WebEventManager webeventManager = new WebEventManager();
            long calendarId = Convert.ToInt64(uxWebCalendarID.Text);
            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;
            String et = uxEndDate.Text + " " + uxEndTime.SelectedItem.Text + ":00:00 " +
```

```
uxEndPeriod.SelectedItem.Text;
            DateTime startdatetime = DateTime.Parse(st);
            DateTime endDatetime = DateTime.Parse(et);
            //Check the WebEvent Details
            List<WebEventData> webeventList = webeventManager.GetNonVariantEventList
(calendarId, startdatetime, endDatetime);
            if (webeventList != null)
            {
                //Display the Web Event Details
                uxEventlistView.Visible = true;
                uxEventlistView.DataSource = webeventList;
                uxEventlistView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Calendar
ID and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetNonVariantEventList (ID)

```
GetNonVariantEventList(System.Int64)
```

Retrieves list of original `eventdata` items for the calendar. For recurring events, this method returns only the actual event items and not the occurrences or variants of recurring events.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID

Parameters

- `calendarId`. ID of the calendar for which to retrieve events.

Returns

Retrieved list of WebEventData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebCalendarIDLabel" AssociatedControlID="uxWebCalendarID"
CssClass="span-4 last"
            runat="server" Text="*WebCalendar ID :" />
        <ektronUI:TextField ID="uxWebCalendarID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="88" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix"></li>
    <asp:ListView ID="uxEventlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
        Visible="false">
        <EmptyDataTemplate>
            Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
                        <th>
                            WebEvent Id
                        </th>
                        <th>
                            Title
                        </th>
                        <th>
                            Location
                        </th>
                        <th>
                            Start Date
                        </th>
                        <th>
                            End Date
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
```

```
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("Id")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Title")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Location")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventStart")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventEnd")%>
                </td>
            </tr>
        </ItemTemplate>
    </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            WebEventManager webeventManager = new WebEventManager();
            long calendarId = Convert.ToInt64(uxWebCalendarID.Text);
            //Check the WebEvent Details
            List<WebEventData> webeventList = webeventManager.GetNonVariantEventList
(calendarId);
            if (webeventList != null)
            {
                //Display the Web Event Details
                uxEventlistView.Visible = true;
                uxEventlistView.DataSource = webeventList;
                uxEventlistView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Calendar
```

```
ID and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }


    }
```

GetVariantEventList (date/time)

```
GetVariantEventList(System.Int64,System.DateTime,System.DateTime)
```

Retrieves an `EventVarianceDictionary` containing all the variances for the supplied calendar in the given time frame.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID
- * Start Date Time
- * End Date Time

Parameters

- `calendarId`. ID of the calendar for which to retrieve events.
- `startUtc`. Start of time period to retrieve.
- `endUtc`. End of time period to retrieve.

Returns

Retrieved `WebEventVarianceDictionary` object which is a specialized Dictionary class for organizing event variances. Key is the recurring event ID. Value is collection of WebEvents that are variances associated with the original recurring event.

```
public class WebEventVarianceDictionary :
  System.Collections.Generic.Dictionary<long,List<WebEventData>>
```

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebCalendarIDLabel" AssociatedControlID="uxWebCalendarID"
CssClass="span-4 last"
            runat="server" Text="*WebCalendar ID :" />
        <ektronUI:TextField ID="uxWebCalendarID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="88" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-4 last"
            runat="server" Text="*Start DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEndDateLabel" AssociatedControlID="uxEndDate"
CssClass="span-4 last"
            runat="server" Text="*End DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxEndDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxEndTime" runat="server">
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
```

```
            <asp:ListItem>12</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEndPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix"></li>
    <asp:ListView ID="uxEventlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
        Visible="false">
        <EmptyDataTemplate>
            Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
                        <th>
                            WebEvent Id
                        </th>
                        <th>
                            Title
                        </th>
                        <th>
                            Location
                        </th>
                        <th>
                            Start Date
                        </th>
                        <th>
                            End Date
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("Id")%>
                </td>
                <td class="devsite-method">
```

```
                    <%# Eval("Title")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Location")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventStart")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventEnd")%>
                </td>
            </tr>
        </ItemTemplate>
    </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            WebEventManager webeventManager = new WebEventManager();
            long calendarId = Convert.ToInt64(uxWebCalendarID.Text);
            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;
            String et = uxEndDate.Text + " " + uxEndTime.SelectedItem.Text + ":00:00 " +
uxEndPeriod.SelectedItem.Text;
            DateTime startdatetime = DateTime.Parse(st);
            DateTime endDatetime = DateTime.Parse(et);
            //Check the WebEvent Details
            WebEventVarianceDictionary webEventVarianceDictionary =
webeventManager.GetVarianceEventList(calendarId, startdatetime, endDatetime);
            List<WebEventData> webEventData = new List<WebEventData>();
            if (webEventVarianceDictionary != null)
            {
                if (webEventVarianceDictionary.Count > 0)
                {
                    foreach (KeyValuePair<long, List<WebEventData>> key in
webEventVarianceDictionary)
                    {
                        webEventData.AddRange(key.Value);
```

```
                    }

                }
                //Display the Web Event Details
                uxEventlistView.Visible = true;
                uxEventlistView.DataSource = webEventData;
                uxEventlistView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Calendar
ID and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetVariantEventList (ID)

```
GetVariantEventList(System.Int64)
```

Retrieves an `EventVarianceDictionary` containing all the variances for the supplied calendar in the given time frame.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Calendar ID

Parameters

- `calendarId`. ID of the calendar for which to retrieve events.

Returns

Retrieved `WebEventVarianceDictionary` object which is a specialized Dictionary class for organizing event variances. Key is the recurring event ID. Value is collection of WebEvents that are variances associated with the original recurring event.

```
public class WebEventVarianceDictionary :
  System.Collections.Generic.Dictionary<long,List<WebEventData>>
```

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebCalendarIDLabel" AssociatedControlID="uxWebCalendarID"
CssClass="span-4 last"
            runat="server" Text="*WebCalendar ID :" />
        <ektronUI:TextField ID="uxWebCalendarID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="88" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix"></li>
    <asp:ListView ID="uxEventlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
        Visible="false">
        <EmptyDataTemplate>
            Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
                        <th>
                            WebEvent Id
                        </th>
                        <th>
                            Title
                        </th>
                        <th>
                            Location
                        </th>
                        <th>
                            Start Date
                        </th>
                        <th>
                            End Date
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
```

```
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("Id")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Title")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Location")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventStart")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("EventEnd")%>
                </td>
            </tr>
        </ItemTemplate>
    </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            WebEventManager webeventManager = new WebEventManager();
            long calendarId = Convert.ToInt64(uxWebCalendarID.Text);
            //Check the WebEvent Details
            WebEventVarianceDictionary webEventVarianceDictionary =
webeventManager.GetVarianceEventList(calendarId);
            List<WebEventData> webEventData = new List<WebEventData>();

            if (webEventVarianceDictionary != null)
            {
                if (webEventVarianceDictionary.Count > 0)
                {
                    foreach (KeyValuePair<long, List<WebEventData>> key in
webEventVarianceDictionary)
                    {
                        webEventData.AddRange(key.Value);
```

```
                    }

                }
                //Display the Web Event Details
                uxEventlistView.Visible = true;
                uxEventlistView.DataSource = webEventData;
                uxEventlistView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,"Please Enter Valid Calendar ID
and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.WebEventData)
```

Updates an existing WebEvent item in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Web Event ID
- Web Event Title
- Location
- Description
- Start Date Time
- End Date Time

Parameters

- `eventData`. `WebEventData` object to update.

Returns

Returns the WebEventData object updated.

Remarks

Validate the WebEvent item with `GetItem()` before you update the properties. Then, call `Update` with your modified `WebEventData` object.

NOTE: You cannot change all properties after the initial Add event, such as the WebEventData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWebEventIDLabel" AssociatedControlID="uxWebEventID"
CssClass="span-4 last"
            runat="server" Text="* WebEventID :" />
        <ektronUI:TextField ID="uxWebEventID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="250" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxWebEventTitleLabel" AssociatedControlID="uxWebEventTitle"
CssClass="span-4 last"
            runat="server" Text=" WebEvent Title :" />
        <ektronUI:TextField ID="uxWebEventTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLocationLabel" AssociatedControlID="uxLocation"
CssClass="span-4 last"
            runat="server" Text="Location :" />
        <ektronUI:TextField ID="uxLocation" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxWebEventDescriptionLabel"
AssociatedControlID="uxWebEventDescription"
            CssClass="span-4 last" runat="server" Text="Description :" />
        <asp:TextBox TextMode="MultiLine" Rows="3" ID="uxWebEventDescription"
CssClass="span-4"
            runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-4 last"
            runat="server" Text=" Start DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
```

```
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEndDateLabel" AssociatedControlID="uxEndDate"
CssClass="span-4 last"
            runat="server" Text=" End DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxEndDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxEndTime" runat="server">
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
            <asp:ListItem>12</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEndPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Framework.Calendar;
using Ektron.Cms.Common.Calendar;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            WebEventManager webeventManager = new WebEventManager();
            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;
            String et = uxEndDate.Text + " " + uxEndTime.SelectedItem.Text + ":00:00 " +
uxEndPeriod.SelectedItem.Text;
            DateTime startdatetime = DateTime.Parse(st);
            DateTime endDatetime = DateTime.Parse(et);
            long eventid = Convert.ToInt64(uxWebEventID.Text);

            //Check the WebEvent Details
            WebEventData WebeventData = webeventManager.GetItem(eventid);
            if (WebeventData != null)
            {
                WebeventData.Title = uxWebEventTitle.Text != string.Empty ?
uxWebEventTitle.Text : WebeventData.Title;
                WebeventData.Description = uxWebEventDescription.Text != string.Empty ?
uxWebEventDescription.Text : WebeventData.Description;
                WebeventData.Location = uxLocation.Text != string.Empty ?
uxLocation.Text : WebeventData.Location;
                if (!string.IsNullOrEmpty(uxStartDate.Text)) { WebeventData.EventStart =
startdatetime; }
                if (!string.IsNullOrEmpty(uxEndDate.Text)) { WebeventData.EventEnd =
endDatetime; }
                WebeventData.DisplayTitle = uxWebEventTitle.Text != string.Empty ?
uxWebEventTitle.Text : WebeventData.DisplayTitle;
                //Update a webEvent with given details.
                webeventManager.Update(WebeventData);
                MessageUtilities.UpdateMessage(uxMessage, "Web Event Details has been
Updated for Web Event ID : " + WebeventData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Web
Calendar Event ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

Data Classes

EventTaxonomyCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms
```

Constructors

- `AddFilter(string, [bool])`

```
public Ektron.Cms.EventTaxonomyFilterGroup
  AddFilter(string taxonomyPath, [bool recursive = false])
```

- `AddFilter(long, [bool])`

```
public Ektron.Cms.EventTaxonomyFilterGroup
  AddFilter(long taxonomyId, [bool recursive = false])
```

- `AddFilter(Ektron.Cms.Common.EventTaxonomyProperty, Ektron.Cms.Common.CriteriaFilterOperator, object, [bool])`

```
public Ektron.Cms.EventTaxonomyFilterGroup
  AddFilter(Ektron.Cms.Common.EventTaxonomyProperty taxonomyProperty,
  Ektron.Cms.Common.CriteriaFilterOperator filterOperator, object value,
  [bool recursive = false])
```

- `EventTaxonomyCriteria()`

```
 public EventTaxonomyCriteria()
```

- `EventTaxonomyCriteria(Ektron.Cms.Common.EventProperty, EkEnumeration.OrderByDirection)`

```
public EventTaxonomyCriteria(Ektron.Cms.Common.EventProperty orderByField,
  EkEnumeration.OrderByDirection orderByDirection)
```

- `GenerateSql(System.Data.Common.DbCommand, System.Collections.Generic.Dictionary<Ektron.Cms.Common.EventProperty,string>, out string, out string)`

```
public override void GenerateSql(System.Data.Common.DbCommand command,
  System.Collections.Generic.Dictionary<EventProperty,string> columnMap,
  out string whereClause, out string orderByClause)
```

- `ToCacheKey()`

```
public override string ToCacheKey()
```

- `TaxonomyGroupFilters`

```
public System.Collections.Generic.List<EventTaxonomyFilterGroup>
  TaxonomyGroupFilters { set; get; }
```

WebEventCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Common
```

Constructors

- `WebEventCriteria()`

  ```
  public WebEventCriteria()
  ```

- `WebEventCriteria(Ektron.Cms.Common.WebEventProperty, EkEnumeration.OrderByDirection)`

  ```
  public WebEventCriteria(Ektron.Cms.Common.WebEventProperty orderByField,
    EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `WebEventProperty` are:

  - CalendarId
  - DateCreated
  - DateModified
  - Description
  - Duration
  - EndUtc
  - Id
  - IsAllDay
  - IsCancelled
  - IsVariance
  - LanguageId
  - LastEditorFirstName
  - LastEditorLastName
  - Location
  - ParentEventId
  - RecurrenceEndDate
  - RecurrenceId
  - RecurrenceType
  - StartUtc
  - Status
  - Title

WebEventData

Namespace

```
Ektron.Cms.Common.Calendar
```

Properties

- `CanDelete`. Indicates whether the current user has permission to delete the event. It is filled only when loaded from `GetWebEventList`.

```
public bool CanDelete { set; get; }
```

- `CanEdit`. Indicates whether the current user has permission to edit the event. It is filled only when loaded from `GetWebEventList`.

```
public bool CanEdit { set; get; }
```

- `Clone()`. Returns a clone of the current `WebEventData` object.

```
public virtual Ektron.Cms.Common.Calendar.WebEventData Clone()
```

- `CreateVariance(System.DateTime)`. Creates a variance event for the current recurring event. Returns an unsaved `WebEvent` representing the event variance. `occurrenceStartUtc` is the event occurrence date for which to create a variance.

```
public virtual Ektron.Cms.Common.Calendar.WebEventData
 CreateVariance(System.DateTime occurrenceStartUtc)
```

- `CreateVariance(System.DateTime, System.DateTime)`. Creates a variance event for the current recurring event. Returns an unsaved `WebEvent` representing the event variance. `occurrenceStartUtc` is the event occurrence date for which to create a variance. `newStartUtc` is the new occurrence start date (UTC).

```
public virtual Ektron.Cms.Common.Calendar.WebEventData
  CreateVariance(System.DateTime originalStartUtc, System.DateTime newStartUtc)
```

- `CurrentEditor`. Filled with the current editor if there is one.

```
public long CurrentEditor { set; get; }
```

- `CurrentTimeZone`. Gets or sets the current timezone for the event. Setting the timezone will alter the value of the `EventStart` property based upon the new timezone's UTC Bias. The underlying UTC value of this event's eventstart property will not change. However, any new values applied to `EventStart` will use the new TimeZone when setting the underlying `EventStartUtc` value.

```
public Ektron.Cms.Common.Calendar.TimeZoneInfo CurrentTimeZone { set; get; }
```

- `Description`. Gets or sets the description of the event.

```
public string Description { set; get; }
```

- `DisplayTitle`. Gets or sets the display title of event. This could be different than the base `Content.Title` if this is a variant of a recurring event. In that case the `Content.Title` may be "Event Name (2)" and the `DisplayTitle` would still be "Event Name".

```
public string DisplayTitle { set; get; }
```

- `Duration`. Gets or sets the current duration of the event. Changing this will affect the `EventEnd` time.

```
public System.TimeSpan Duration { set; get; }
```

- `EventEnd`. Gets and sets the event's end time based upon the current timezone. Setting this value will alter `EventEndUTC` based upon the current timezone's UTC

bias.

```
public System.DateTime EventEnd { set; get; }
```

- `EventEndUtc`. Gets or sets the `EventEnd` time of the event in UTC time.

```
public System.DateTime EventEndUtc { set; get; }
```

- `EventStart`. Gets and sets the event's start time based upon the current timezone. Setting this value will alter `EventStartUTC` based upon the current timezone's UTC bias.

```
public System.DateTime EventStart { set; get; }
```

- `EventStartUtc`. Gets or sets the `EventStart` time of the event in UTC time.

```
public System.DateTime EventStartUtc { set; get; }
```

- `GetEventStartUtcForOccurrence(System.DateTime)`. Returns the `EventStartUtc` for a specific event occurrence. This method is only applicable to recurring events. The `EventStartUtc` time is based upon the `EventStart` for originating timezone and may fluctuate. The local `EventStart` for the originating timezone will always remain consistent. For example, an 8:00 AM ET is consistent in both standard and daylight time and therefore, the UTC value may change based upon the occurrence date.

```
public System.DateTime GetEventStartUtcForOccurrence(System.DateTime
occurrenceDate)
```

- `GetEventStartUtcForOccurrence (Ektron.Cms.Common.Calendar.WebEventData, System.DateTime)`. Returns the `EventStartUtc` for event occurrence on the supplied date. This method is only applicable to recurring events. The `EventStartUtc` time is based upon the `EventStart` for originating timezone and may fluctuate. The local `EventStart` for the originating timezone will always remain consistent. For example, an 8:00 AM ET is consistent in both standard and daylight time and therefore, the UTC value may change based upon the occurrence date. Exposing this statically so the logic can be used by recurrence managers without creating instance of `WebEvent` as they only require `WebEventData` classes.

```
public static System.DateTime
  GetEventStartUtcForOccurrence(Ektron.Cms.Common.Calendar.WebEventData eventData,
  System.DateTime occurrenceDate)
```

- `GetOriginalTimeZoneEventStart()`. Returns the recurring `EventStart` in the originating timezone. This time should remain consistent for all occurrences. This method is only applicable to recurring events.

```
public System.DateTime GetOriginalTimeZoneEventStart()
```

- `GetOriginalTimeZoneEventStart (Ektron.Cms.Common.Calendar.WebEventData)`. Returns the recurring `EventStart` in the originating timezone. This time should remain consistent for all occurrences. This method is only applicable to recurring events. Exposing this statically so the logic can be used by recurrence managers without creating instance of `WebEvent` as they only require `WebEventData` classes.

```
public static System.DateTime
 GetOriginalTimeZoneEventStart(Ektron.Cms.Common.Calendar.WebEventData eventData)
```

- `IsAllDay`. Gets or sets the `IsAllDay` flag of the event. A true value means the event is an all day event with no actual start\end.

```
public bool IsAllDay { set; get; }
```

- `IsCancelled`. Gets or sets the `IsVariance` flag for the event. A true value means that this event is a cancelled occurrence of a recurring event.

```
public bool IsCancelled { set; get; }
```

- `IsRecurring`. Gets a boolean value indicating if the current event is a recurring event.

```
public bool IsRecurring { get; }
```

- `IsVariance`. Gets or sets the `IsVariance` flag for the event. A true value means this event is a variance of an occurrence of a recurring event.

```
public bool IsVariance { set; get; }
```

- `Location`. Gets or sets the location of the event.

```
public string Location { set; get; }
```

- `OriginalStartUtc`. Gets or sets the `OriginalStartUtc` time. The original `OriginalStartUtc` is the UTC time of the original occurrence of a recurring event. It is used to calculate TimeZone adjusted times of event occurrences.

```
public System.DateTime OriginalStartUtc { set; get; }
```

- `OriginalTimeZone`. Gets or sets the original timezone for this event

```
public Ektron.Cms.Common.Calendar.TimeZoneInfo OriginalTimeZone { set; get; }
```

- `ParentEventId`. Gets or sets the recurrence ID of the event. The recurrence ID is the event ID of the original recurring event. This value is needed to tie event variances back to the original recurring event.

```
public long ParentEventId { set; get; }
```

- `Recurrence`. Gets or sets the current recurrence pattern for the event.

```
public Ektron.Cms.Common.RecurrenceData Recurrence { set; get; }
```

- `UtcOffset`. Gets the `UtcOffset` of the current timezone.

```
public System.TimeSpan UtcOffset { get; }
```

- `Validate()`. Validates the event object and returns a list of errors.

```
public virtual ValidationResults Validate()
```

- `WebEventData()`

```
public WebEventData()
```

Commerce

9.00 and higher

The Commerce category manages commerce exchange on the Web and has the following classes:

- *AddressManager* on the next page. Manages addresses in the CMS.
- *BasketItemManager* on page 345. Manages product items in the basket.
- *BasketManager* on page 360. Manipulates the baskets in e-commerce.
- *CatalogEntryManager* on page 387. Manipulates the catalog entry data in CMS.
- *CountryManager* on page 419. Manipulates the country data in the CMS.
- *CouponManager* on page 435. Manipultes the coupon data in the CMS. It can be used for add, edit, and get the coupon data object.
- *CreditCardTypeManager* on page 480. Manages credit card types in the CMS.
- *CurrencyManager* on page 499. Manages commerce currencies in the CMS.
- *CustomerManager* on page 519. Manipulates e-commerce customer details in CMS.
- *ExchangeRateManager* on page 550. Manages commerce exchange rates in the CMS.
- *InventoryManager* on page 570. Manages inventory in the CMS.
- *OrderManager* on page 591. Manages commerce orders in the CMS.
- *PackageManager* on page 628. Manages commerce packages in the CMS.
- *PasswordHistoryManager* on page 644. Manages password histories.
- *PaymentGatewayManager* on page 665. Manages the commerce paymentgateway in the CMS.
- *ProductTypeManager* on page 683. Manipulates the catalog's product type data in CMS.
- *RecommendationManager* on page 700. Manipulates the recommendated items to the catalog entry.
- *RegionManager* on page 715. Manages regions in the CMS.
- *ShippingMethodManager* on page 730. Manages shipping methods in CMS.
- *TaxClassManager* on page 743. Manages tax class in the CMS.
- *TaxRateManager* on page 755. Manages commerce tax rates in the CMS.
- *TaxTypeManager* on page 780. Manages commerce tax types in the CMS.
- *WarehouseManager* on page 792. Manages commerce warehouses in the CMS.

AddressManager

9.00 and higher

The class manages addresses in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.AddressManager
```

Constructors

- `AddressManager()`
- `AddressManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add (address)

```
Add(Ektron.Cms.Commerce.AddressData)
```

Adds a new address based on information in an AddressData object. AddressData.Id is populated with the newly created ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Address Name
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID
- * Phone

Parameters

- `addressData`. The AddressData object to add.

Returns

Returns added AddressData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressNameLabel" AssociatedControlID="uxAddressName"
CssClass="span-5 last" runat="server" Text="* AddressName :" />
        <ektronUI:TextField ID="uxAddressName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressLine1Label" AssociatedControlID="uxAddressLine1"
CssClass="span-5 last" runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxAddressLine1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCityLabel" AssociatedControlID="uxCity" CssClass="span-5
last" runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLabel" AssociatedControlID="uxRegionId"
CssClass="span-5 last" runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPostalCodeLabel" AssociatedControlID="uxPostalCode"
CssClass="span-5 last" runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxShippingFromPostalCode" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIdLabel" AssociatedControlID="uxCountryId"
CssClass="span-5 last" runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPhoneLabel" AssociatedControlID="uxPhone" CssClass="span-5
last" runat="server" Text="* Phone :" />
        <ektronUI:TextField ID="uxPhone" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            AddressManager addressManager = new AddressManager();

            AddressData addressData = new AddressData();
            addressData.Name = uxAddressName.Text;
            addressData.AddressLine1 = uxAddressLine1.Text;
            addressData.City = uxCity.Text;
            addressData.Region = new RegionData { Id = long.Parse(uxRegionId.Text) };
            addressData.PostalCode = uxPostalCode.Text;
            addressData.Phone = uxPhone.Text;
            addressData.Country = new CountryData { Id = int.Parse(uxCountryId.Text) };

            addressManager.Add(addressData);
```

```
            MessageUtilities.UpdateMessage(uxMessage, "AddressData is added with Id " +
addressData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

Add (customer ID)

```
Add(Ektron.Cms.Commerce.AddressData,System.Int64)
```

Adds a new address based on information in an AddressData object and associates with a customer. AddressData.Id is populated with the newly created ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Address Name
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID
- * Phone
- * Customer ID

Parameters

- `addressData`. The AddressData object to add.
- `customerId`. The ID of customer to which this address belongs.

Returns

Returns added AddressData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressNameLabel" AssociatedControlID="uxAddressName"
```

```
CssClass="span-5 last" runat="server" Text="* AddressName :" />
        <ektronUI:TextField ID="uxAddressName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressLine1Label" AssociatedControlID="uxAddressLine1"
CssClass="span-5 last" runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxAddressLine1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCityLabel" AssociatedControlID="uxCity" CssClass="span-5
last" runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLabel" AssociatedControlID="uxRegionId"
CssClass="span-5 last" runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPostalCodeLabel" AssociatedControlID="uxPostalCode"
CssClass="span-5 last" runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxShippingFromPostalCode" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIdLabel" AssociatedControlID="uxCountryId"
CssClass="span-5 last" runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPhoneLabel" AssociatedControlID="uxPhone" CssClass="span-5
last" runat="server" Text="* Phone :" />
        <ektronUI:TextField ID="uxPhone" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCustomerIdLabel" AssociatedControlID="uxCustomerId"
CssClass="span-5 last" runat="server" Text="* CustomerId :" />
        <ektronUI:TextField ID="uxCustomerId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            AddressManager addressManager = new AddressManager();
            long customerId = long.Parse(uxCustomerId.Text);
            AddressData addressData = new AddressData();
            addressData.Name = uxAddressName.Text;
            addressData.AddressLine1 = uxAddressLine1.Text;
            addressData.City = uxCity.Text;
            addressData.Region = new RegionData { Id = long.Parse(uxRegionId.Text) };
            addressData.PostalCode = uxPostalCode.Text;
            addressData.Phone = uxPhone.Text;
            addressData.Country = new CountryData { Id = int.Parse(uxCountryId.Text) };

            long id=addressManager.Add(addressData, customerId).Id;
            MessageUtilities.UpdateMessage(uxMessage, "Customer AddressData is added
with Id " + id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

Deletes an address.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `addressId`. The ID of the AddressData object to delete.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressIdLabel" AssociatedControlID="uxAddressId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxAddressId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxAddressId.Text);
            AddressManager addressManager = new AddressManager();
            addressManager.Delete(ID);
            MessageUtilities.UpdateMessage(uxMessage, "AddresssData with Id " +
uxAddressId.Text + " is deleted", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves an AddressData object by providing its ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `addressId`. The ID the of AddressData object to retrieve.

Returns

Returns an AddressData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressIdLabel" AssociatedControlID="uxAddressId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxAddressId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
    </ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAddressName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAddressLine1" runat="server"></asp:Literal>
    </li>
        <li class="clearfix">
        <asp:Literal ID="uxCity" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxRegionId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
```

```
        <asp:Literal ID="uxPostalCode" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCountryId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxPhone" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long addressId = long.Parse(uxAddressId.Text);

            AddressManager addressManager = new AddressManager();
            AddressData addressData = new AddressData();
            addressData = addressManager.GetItem(addressId);
            if (addressData != null)
            {

                MessageUtilities.UpdateMessage(uxMessage, "Details for address Id " +
addressData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxAddressName.Text = "Address Name : " + addressData.Name;
                uxAddressLine1.Text = "AddressLine1 : " + addressData.AddressLine1;
                uxCity.Text = "City : " + addressData.City;
                uxRegionId.Text = "RegionId : " + addressData.Region.Id;
                uxPostalCode.Text = "PostalCode : " + addressData.PostalCode;
                uxCountryId.Text = "CountryId : " + addressData.Country.Id;
                uxPhone.Text = "Phone : " + addressData.Phone;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either address with ID " +
addressId + " does not exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
```

```
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.AddressCriteria)
```

Retrieves the list of addresses based on the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Address Property
- * Object Value

Parameters

- `criteria`. Criteria used to retrieve the address.

Returns

Returns the list of AddressData object.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressPropertyLabel"
AssociatedControlID="uxAddressProperty" CssClass="span-5 last" runat="server" Text="*
Address Property:" />
        <asp:DropDownList ID="uxAddressProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
            <asp:ListItem>City</asp:ListItem>
            <asp:ListItem>RegionId</asp:ListItem>
            <asp:ListItem>CountryId</asp:ListItem>
            <asp:ListItem>PostalCode</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last" runat="server" Text="* ObjectValue:" />
```

```
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxAddressDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >No records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>Id</th>
                    <th>Name</th>
                    <th>AddressLine1</th>
                    <th>City</th>
                    <th>PostalCode</th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("AddressLine1")%>
            </td>
             <td class="devsite-method">
                <%# Eval("City")%>
            </td>
             <td class="devsite-method">
                <%# Eval("PostalCode")%>
            </td>

      </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            AddressManager addressManager = new AddressManager();
            AddressCriteria criteria = new AddressCriteria();

            if (uxAddressProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(AddressProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else if (uxAddressProperty.SelectedItem.Text == "Name")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(AddressProperty.Name, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else if (uxAddressProperty.SelectedItem.Text == "City")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(AddressProperty.City, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else if (uxAddressProperty.SelectedItem.Text == "RegionId")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(AddressProperty.RegionId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxAddressProperty.SelectedItem.Text == "CountryId")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(AddressProperty.CountryId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
```

```
                criteria.AddFilter(AddressProperty.PostalCode,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            List<AddressData> addressDataList = addressManager.GetList(criteria);

            uxAddressDataListView.Visible = true;
            uxAddressDataListView.DataSource = addressDataList;
            uxAddressDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.AddressData)
```

Updates the supplied Address.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Id
- * Address Name
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID
- * Phone
- * Customer ID

Parameters

- `addressData`. The ID the of AddressData object to retrieve.

Returns

Returns an updated AddressData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressIdLabel" AssociatedControlID="uxAddressId"
CssClass="span-5 last" runat="server" Text=" Id:" />
        <ektronUI:TextField ID="uxAddressId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressNameLabel" AssociatedControlID="uxAddressName"
CssClass="span-5 last" runat="server" Text="* AddressName :" />
        <ektronUI:TextField ID="uxAddressName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressLine1Label" AssociatedControlID="uxAddressLine1"
CssClass="span-5 last" runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxAddressLine1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCityLabel" AssociatedControlID="uxCity" CssClass="span-5
last" runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLabel" AssociatedControlID="uxRegionId"
CssClass="span-5 last" runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPostalCodeLabel" AssociatedControlID="uxPostalCode"
CssClass="span-5 last" runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxPostalCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIdLabel" AssociatedControlID="uxCountryId"
CssClass="span-5 last" runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPhoneLabel" AssociatedControlID="uxPhone" CssClass="span-5
last" runat="server" Text="* Phone :" />
        <ektronUI:TextField ID="uxPhone" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long addressId = long.Parse(uxAddressId.Text);
            AddressManager addressManager = new AddressManager();
            AddressData addressData = new AddressData();
            addressData = addressManager.GetItem(addressId);
            if (addressId > 0 && addressData != null)
            {
                addressData.Name = uxAddressName.Text;
                addressData.AddressLine1 = uxAddressLine1.Text;
                addressData.City = uxCity.Text;
                addressData.Region = new RegionData { Id = long.Parse(uxRegionId.Text)
};
                addressData.PostalCode = uxPostalCode.Text;
                addressData.Phone = uxPhone.Text;
                addressData.Country = new CountryData { Id = int.Parse(uxCountryId.Text)
};
                addressManager.Update(addressData);
                MessageUtilities.UpdateMessage(uxMessage, "AddressData is Updated with
Id " + uxAddressId.Text, Message.DisplayModes.Success);

            }
            else
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid address
ID.", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
```

```
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

UpdateOrderAddress

```
UpdateOrderAddress(Ektron.Cms.Commerce.AddressData,System.Int64,
  System.Boolean,System.Boolean)
```

Updates an address associated with an order. This update only affects this specific address and does not affect the customer's actual saved address.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- ID
- * Address Name
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID
- * Phone
- * Order ID
- * Is Shipping
- * Is Billing

Parameters

- `addressData`. Address information to update.
- `orderId`. ID of the order for which to update the address.
- `isBilling`. If true, updates the billing address for an order.
- `isShipping`. If true, updates the shipping address for an order.

Returns

Returns an updated AddressData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressIdLabel" AssociatedControlID="uxAddressId"
CssClass="span-5 last" runat="server" Text=" Id:" />
        <ektronUI:TextField ID="uxAddressId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressNameLabel" AssociatedControlID="uxAddressName"
CssClass="span-5 last" runat="server" Text="* AddressName :" />
        <ektronUI:TextField ID="uxAddressName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressLine1Label" AssociatedControlID="uxAddressLine1"
CssClass="span-5 last" runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxAddressLine1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCityLabel" AssociatedControlID="uxCity" CssClass="span-5
last" runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLabel" AssociatedControlID="uxRegionId"
CssClass="span-5 last" runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPostalCodeLabel" AssociatedControlID="uxPostalCode"
CssClass="span-5 last" runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxPostalCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIdLabel" AssociatedControlID="uxCountryId"
CssClass="span-5 last" runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPhoneLabel" AssociatedControlID="uxPhone" CssClass="span-5
last" runat="server" Text="* Phone :" />
        <ektronUI:TextField ID="uxPhone" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-5 last" runat="server" Text="* OrderId :" />
        <ektronUI:TextField ID="uxOrderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsShippingLabel" AssociatedControlID="uxIsShipping"
CssClass="span-5 last" runat="server" Text="* IsShipping: " />
            <asp:checkbox ID="uxIsShipping" runat="server" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsBillingLabel" AssociatedControlID="uxIsBilling"
CssClass="span-5 last" runat="server" Text="* IsBilling: " />
            <asp:checkbox ID="uxIsBilling" runat="server" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="UpdateOrderAddress"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long orderId=long.Parse(uxOrderId.Text);
            long addressId = long.Parse(uxAddressId.Text);
            AddressManager addressManager = new AddressManager();
            AddressData addressData = new AddressData();
            addressData = addressManager.GetItem(addressId);
            if (addressId > 0 && addressData != null)
            {
                addressData.Name = uxAddressName.Text;
                addressData.AddressLine1 = uxAddressLine1.Text;
                addressData.City = uxCity.Text;
                addressData.Region = new RegionData { Id = long.Parse(uxRegionId.Text)
};
                addressData.PostalCode = uxPostalCode.Text;
                addressData.Phone = uxPhone.Text;
                addressData.Country = new CountryData { Id = int.Parse(uxCountryId.Text)
};
```

```
                addressManager.UpdateOrderAddress(addressData,
orderId,uxIsShipping.Checked,uxIsBilling.Checked);
                MessageUtilities.UpdateMessage(uxMessage, "Order AddressData is Updated
with Id " + uxAddressId.Text, Message.DisplayModes.Success);

            }
            else
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid address
ID.", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

AddressCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `AddressCriteria()`

  ```
  public AddressCriteria()
  ```

- `AddressCriteria(Ektron.Cms.Common.AddressProperty, EkEnumeration.OrderByDirection)`

  ```
  public AddressCriteria(Ektron.Cms.Common.AddressProperty orderByField,
    EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `AddressProperty` are:

  - City
  - Company
  - CountryId
  - CustomerId
  - Id
  - IsCommercial
  - Name
  - PostalCode

- RegionId
- Validated

AddressData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `AddressData()`. Constructor

```
public AddressData()
```

- `AddressData(string, string, Ektron.Cms.Commerce.RegionData, Ektron.Cms.Commerce.CountryData, string, string)`. Constructor.

```
public AddressData(string name, string addressLine1,
  Ektron.Cms.Commerce.RegionData region,
  Ektron.Cms.Commerce.CountryData country,
  string city, string postalcode)
```

- `AddressData(string, string, Ektron.Cms.Commerce.RegionData, Ektron.Cms.Commerce.CountryData, string, string)`. Constructor.

```
public AddressData(string name, string addressLine1,
  Ektron.Cms.Commerce.RegionData region,
  Ektron.Cms.Commerce.CountryData country,
  string city, string postalCode, bool iscommercial)
```

- `AddressLine1`. Gets or sets the first line of the address. This property is required when creating an address data object.

```
public string AddressLine1 { set; get; }
```

- `AddressLine2`. Gets or sets line 2 of address.

```
public string AddressLine2 { set; get; }
```

- `City`. Gets or sets the city associated with this address. This property is required when creating an address data object.

```
public string City { set; get; }
```

- `Company`. Gets or sets the name of the company associated with this address. For example, if the address is for Ektron, enter "Ektron, Inc." in the property.

```
public string Company { set; get; }
```

- `Country`. Gets or sets the country associated with this address.

```
public Ektron.Cms.Commerce.CountryData Country { set; get; }
```

- `Id`. Gets or sets the ID of an address.

```
public long Id { set; get; }
```

- `IsCommercial`. Gets or sets whether this address is commercial.

```
public bool IsCommercial { set; get; }
```

- `IsValidated`. Gets or sets whether this address is validated. Validation can come from third-party services.

```
public bool IsValidated { set; get; }
```

- `Name`. Gets or sets the name associated with the address. For example, if you are entering an address for Ivan Tory, enter the name "Ivan Tory" in this property. This property is required when creating an address data object.

```
public string Name { set; get; }
```

- `Phone`. Gets or sets the phone number associated with this address.

```
public string Phone { set; get; }
```

- `PostalCode`. Gets or sets the postal code associated with this address. This property is required when creating an address data object.

```
public string PostalCode { set; get; }
```

- `Region`. Gets or sets the region associated with this address.

```
public Ektron.Cms.Commerce.RegionData Region { set; get; }
```

- `Validate()`

```
public ValidationResults Validate()
```

BasketItemManager

9.00 and higher

The class manages product items in the basket.

Namespace

```
Ektron.Cms.Framework.Commerce.BasketItemManager
```

Constructors

- `BasketItemManager()`
- `BasketItemManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 348
- *GetKitConfiguration(user)* on page 352
- *GetKitConfiguration (shopper)* on page 350
- *Update* on page 354
- *UpdateQuantity* on page 356

Add

```
Add(Ektron.Cms.Commerce.BasketItemData)
```

Adds the product or variant or kit items to the basket.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket ID
- * Product ID
- Quantity

Parameters

- `basketItemData`. BasketItemData object to be added.

Returns

Returns the added basket item data.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-5 last" runat="server" Text="* Basket Id:" />
        <ektronUI:TextField ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxProductIdLabel" AssociatedControlID="uxProductId"
CssClass="span-5 last" runat="server" Text="* Product Id:" />
        <ektronUI:TextField ID="uxProductId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxProductQuantityLabel"
AssociatedControlID="uxProductQuantity" CssClass="span-5 last" runat="server" Text="
Quantity:" />
        <ektronUI:TextField ID="uxProductQuantity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
```

```
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
                        BasketItemManager basketItemManager = new BasketItemManager();
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            long basketId = long.Parse(uxBasketId.Text);
            long productId = long.Parse(uxProductId.Text);
            int quantity= int.Parse(uxProductQuantity.Text);
            BasketData basketData = basketManager.GetItem(basketId);
            if (basketData != null)
            {
                BasketItemData basketItemData = new BasketItemData()
                {
                    BasketId = basketId,
                    ProductId = productId,
                    ProductType =
Ektron.Cms.Common.EkEnumeration.CatalogEntryType.Product,
                    Quantity = quantity
                };

                BasketItemData newBasketItemData = basketItemManager.Add
(basketItemData);
                if (newBasketItemData.Id > 0)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Product " + productId + "
is added with basket " + basketId + " with new basketItem Id " + newBasketItemData.Id,
Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Product adding process
has been failed. ", Message.DisplayModes.Warning);
                }
            }
            else
```

```
            {
                MessageUtilities.UpdateMessage(uxMessage, "Invalid basket Id. ",
Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64,System.Int64)
```

Deletes the basket item from the basket.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket Item ID
- * Basket ID

Parameters

- `basketItemId`. Basket Item ID.
- `basketId`

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxBasketItemIdLabel" AssociatedControlID="uxBasketItemId"
CssClass="span-5 last" runat="server" Text="* BasketItem Id:" />
        <ektronUI:TextField ID="uxBasketItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
   <li class="clearfix">
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-5 last" runat="server" Text="* Basket Id:" />
        <ektronUI:TextField ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
```

```
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            BasketItemManager basketItemManager = new BasketItemManager();
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            long basketId = long.Parse(uxBasketId.Text);
            long basketItemId = long.Parse(uxBasketItemId.Text);

            BasketData basket = basketManager.GetItem(basketId);
            if (basket != null)
            {
                if (basket.Items.FindAll(x => x.Id.Equals(basketItemId)).ToList().Count
> 0)
                {
                    basketItemManager.Delete(basketItemId, basketId);
                    MessageUtilities.UpdateMessage(uxMessage, "Basket item " +
basketItemId + " has been removed from the basket " + basketId,
Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Basket Item Id is not
valid. ", Message.DisplayModes.Warning);
                }
            }
            else
            {
```

```
                MessageUtilities.UpdateMessage(uxMessage, "Basket Id is not valid. ",
Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetKitConfiguration (shopper)

```
GetKitConfiguration(System.Int64,System.String)
```

Retrieves the kit configuration data.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket Item ID
- Shopper ID

Parameters

- `basketItemId`. ID of the basket item.
- `shopperId`. ID of the shopper; it should be a GUID.

Returns

Kit configuration data for the basket item.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketItemIdLabel" AssociatedControlID="uxBasketItemId"
CssClass="span-5 last" runat="server" Text="* BasketItem Id:" />
        <ektronUI:TextField ID="uxBasketItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
     <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxShopperIdLabel" AssociatedControlID="uxShopperId"
```

```
CssClass="span-5 last" runat="server" Text="* Shopper Id:" />
        <ektronUI:TextField ID="uxShopperId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetKitConfigurationForShopper"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxKitConfigId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxBasketItemId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxPriceModifier" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxOptionName" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            BasketItemManager basketItemManager = new BasketItemManager();
            long basketItemId = long.Parse(uxBasketItemId.Text);
            string shopperId = uxShopperId.Text;
```

```
            KitConfigData kitConfigData = basketItemManager.GetKitConfiguration
(basketItemId, shopperId);
            if (kitConfigData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details of the kit
coonfiguration for the basket item " + basketItemId  + " is given below.",
Message.DisplayModes.Success);
                uxKitConfigId.Text = " KitConfig Id : " + kitConfigData.Id;
                uxBasketItemIds.Text = " BasketItem Id : " + kitConfigData.BasketItemId;
                uxPriceModifier.Text = " Price Modifier : " +
kitConfigData.PriceModifier;
                uxOptionName.Text = " Option Name : " + kitConfigData.Groups.Select(x =>
x.Name).FirstOrDefault();

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Basket item id or user id is
not valid. ", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetKitConfiguration(user)

```
GetKitConfiguration(System.Int64,System.Int64)
```

Retrieves the kit configuration data.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket Item ID
- * User ID

Parameters

- `basketItemId`. ID of the basket item.
- `userId`. ID of the user.

Returns

Kit configuration data for the basket item.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketItemIdLabel" AssociatedControlID="uxBasketItemId"
CssClass="span-5 last" runat="server" Text="* BasketItem Id:" />
        <ektronUI:TextField ID="uxBasketItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-5 last" runat="server" Text="* User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetKitConfiguration"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxKitConfigId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxBasketItemId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxPriceModifier" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxOptionName" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            BasketItemManager basketItemManager = new BasketItemManager();

            long basketItemId = long.Parse(uxBasketItemId.Text);
            long userId = long.Parse(uxUserId.Text);

            KitConfigData kitConfigData = basketItemManager.GetKitConfiguration
(basketItemId, userId);
            if (kitConfigData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details of the kit
coonfiguration for the basket item " + basketItemId  + " is given below.",
Message.DisplayModes.Success);
                uxKitConfigId.Text = " KitConfig Id : " + kitConfigData.Id;
                uxBasketItemIds.Text = " BasketItem Id : " + kitConfigData.BasketItemId;
                uxPriceModifier.Text = " Price Modifier : " +
kitConfigData.PriceModifier;
                uxOptionName.Text = " Option Name : " + kitConfigData.Groups.Select(x =>
x.Name).FirstOrDefault();

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Basket item id or user id is
not valid. ", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Update

```
Update(Ektron.Cms.Commerce.BasketItemData)
```

Updates the kit configuration data item in the basket.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Product ID
- * Basket Item ID

Parameters

- `basketItemData`. BasketItemData object to be updated.

Returns

Returns the BasketItemData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxProductIdLabel" AssociatedControlID="uxProductId"
CssClass="span-5 last" runat="server" Text="* Product Id:" />
        <ektronUI:TextField ID="uxProductId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketItemIdLabel" AssociatedControlID="uxBasketItemId"
CssClass="span-5 last" runat="server" Text="* BasketItem Id:" />
        <ektronUI:TextField ID="uxBasketItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
```

```
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            BasketItemManager basketItemManager = new BasketItemManager();
            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            long productId = long.Parse(uxProductId.Text);
            long basketItemId = long.Parse(uxBasketItemId.Text);

            EntryData entryData = new KitData();
            KitData kitData = new KitData();
            entryData = catalogEntryManager.GetItem(productId);
            if ((entryData as KitData) != null)

            {
                kitData = entryData as KitData;
                KitConfigData kitConfigData = new KitConfigData()
                {
                    Groups = kitData != null ? kitData.OptionGroups : new
List<OptionGroupData>()
                };
                BasketItemData basketItemData = new BasketItemData()
                {
                     Id= basketItemId,
                     ProductId= productId,
                     ProductType= Ektron.Cms.Common.EkEnumeration.CatalogEntryType.Kit,
                     Configuration= kitConfigData
                };
                basketItemManager.Update(basketItemData);
               MessageUtilities.UpdateMessage(uxMessage, "Basket Item is updated with
basket item " + basketItemId +".", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "The product is not a Kit. Kit
item can only be updated. Update process is failed. ", Message.DisplayModes.Warning );
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

UpdateQuantity

```
UpdateQuantity(System.Int64,System.Int64,System.Int32)
```

Updates the basket item quantity.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket Item ID
- * Basket ID
- * Quantity

Parameters

- `basketItemId`. ID of the basket item.
- `basketId`. ID of the basket, in which the item quantity is updated.
- `productQuantity`. Product quantity to update.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketItemIdLabel" AssociatedControlID="uxBasketItemId"
CssClass="span-5 last" runat="server" Text="* BasketItem Id:" />
        <ektronUI:TextField ID="uxBasketItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
   <div class="ektronTopSpace"></div>
   <li class="clearfix">
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-5 last" runat="server" Text="* Basket Id:" />
        <ektronUI:TextField ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
   <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxProductQuantityLabel"
AssociatedControlID="uxProductQuantity" CssClass="span-5 last" runat="server" Text="*
Quantity:" />
        <ektronUI:TextField ID="uxProductQuantity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    </li>
    <div class="ektronTopSpace"></div>
```

```
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="UpdateQuantity"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            BasketItemManager basketItemManager = new BasketItemManager();
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            long basketId = long.Parse(uxBasketId.Text);
            long basketItemId = long.Parse(uxBasketItemId.Text);
            int quantity = int.Parse(uxProductQuantity.Text);

            BasketData basket = basketManager.GetItem(basketId);
            if(basket!=null )
            {
                if (basket.Items.FindAll(x => x.Id.Equals(basketItemId)).ToList().Count
> 0)
                {
                    basketItemManager.UpdateQuantity(basketItemId,basketId, quantity);
                    MessageUtilities.UpdateMessage(uxMessage, "Quantity " + quantity + "
is updated to the basket item " + basketItemId + " in the basket " + basketId,
Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "BasketItem Id is not
valid. ", Message.DisplayModes.Warning);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Basket Id is not valid. " ,
Message.DisplayModes.Warning );
```

```
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

BasketItemData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `BasketId`

  ```
  public long BasketId { set; get; }
  ```

- `VariantId`

  ```
  public long VariantId { set; get; }
  ```

BasketManager

9.00 and higher

This class manipulates the baskets in eCommerce.

Namespace

```
Ektron.Cms.Framework.Commerce.BasketManager
```

Constructors

- `BasketManager()`
- `BasketManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *SetDefault* on page 381
- *Update* on page 383

Add

```
Add(Ektron.Cms.Commerce.BasketData)
```

Adds the new basket to the specific user in CMS. The `basketData.Id` is populated with the Id of the new basket.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- Shopper ID

Parameters

- `basketData`. Basket object to be added.

Returns

Returns the added BasketData object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxBasketNameLabel" AssociatedControlID="uxBasketName"
CssClass="span-3 last" runat="server" Text="* Name:" />
        <ektronUI:TextField  ID="uxBasketName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketUserIdLabel" AssociatedControlID="uxBasketUserId"
CssClass="span-3 last" runat="server" Text=" User Id:" />
        <ektronUI:TextField  ID="uxBasketUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxBasketShopperIdLabel"
AssociatedControlID="uxBasketShopperId" CssClass="span-3 last" runat="server" Text="
Shopper Id:" />
        <ektronUI:TextField  ID="uxBasketShopperId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketDefaultLabel" AssociatedControlID="uxBasketDefault"
```

```
CssClass="span-3 last" runat="server" Text=" IsDefault:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxBasketDefault"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            bool isDefault = false;
            if (uxBasketDefault.Checked)
            {
                isDefault = true;
            }
            BasketData basketData = new BasketData()
            {
                Name = uxBasketName.Text,
                UserId = long.Parse(uxBasketUserId.Text),
                ShopperId = uxBasketShopperId.Text,
                IsDefault = isDefault
            };
            BasketData newBasketData = basketManager.Add(basketData);

            if (newBasketData.Id > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Basket is added with Id " +
newBasketData.Id, Message.DisplayModes.Success);
            }
```

```
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Basket adding process is
failed", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

ApplyCoupon

```
ApplyCoupon(System.Int64,System.Int64)
```

Apply the coupon to the basket.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket ID
- * Coupon ID

Parameters

- `basketId`. ID of the basket.
- `couponId`. ID of the coupon to be applied to the basket.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-3 last" runat="server" Text="* Basket Id:" />
        <ektronUI:TextField  ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Apply Coupon"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            CouponManager couponManager = new CouponManager();
            long couponId=long.Parse(uxCouponId.Text);
            long basketId= long.Parse(uxBasketId.Text);
                       CouponData couponData = couponManager.GetItem(couponId);
            BasketData basketData = basketManager.GetItem(basketId);

            if (couponData != null && basketData!=null)
            {
                basketManager.ApplyCoupon(basketId, couponId);
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + "
applied to the basket " + basketId, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Invalid Coupon or Basket Id."
, Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a basket from the CMS by its ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket ID

Parameters

- `basketId`. ID of basket to be deleted.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-4 last" runat="server" Text="* Basket Id:" />
        <ektronUI:TextField ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            long ID = long.Parse(uxBasketId.Text);
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            basketManager.Delete(ID);
            MessageUtilities.UpdateMessage(uxMessage, "BasketData with Id " +
uxBasketId.Text + " is deleted", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Empty

```
Empty(System.Int64)
```

Deletes all the items in a basket by it's basket ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket ID

Parameters

- `basketId`. ID of basket to be deleted.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-4 last" runat="server" Text="* Basket Id:" />
        <ektronUI:TextField ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Empty"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long basketId = long.Parse(uxBasketId.Text);
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            basketManager.Empty(basketId);
            MessageUtilities.UpdateMessage(uxMessage, "Basket items are deleted",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetAppliedCoupons

```
GetAppliedCoupons(System.Int64)
```

Retrieves the list of coupon items which are applied to the basket.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket ID

Parameters

- `basketId`. ID of basket.

Returns

List of coupons.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-4 last" runat="server" Text="* Basket Id:" />
        <ektronUI:TextField ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetAppliedCoupons"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCouponListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Id
                    </th>
                    <th>
                        Code
                    </th>
                    <th>
                        IsActive
                    </th>
                    <th>
                        CurrencyId
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
```

```
                <%# Eval("Code")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsActive")%>
            </td>
            <td class="devsite-method">
                <%# Eval("CurrencyId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            long basketId = long.Parse(uxBasketId.Text);
            List<CouponData> listCouponData = basketManager.GetAppliedCoupons(basketId);

            uxCouponListView.Visible = true;
            uxCouponListView.DataSource = listCouponData;
            uxCouponListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetDefault (default)

```
GetDefault()
```

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Returns

The default basket object.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetDefault"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsDefault" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSubtotal" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            BasketData basketData = new BasketData();
            basketData = basketManager.GetDefault();
            if (basketData != null)
            {

                MessageUtilities.UpdateMessage(uxMessage, "Details for default basket
data are below", Message.DisplayModes.Success);
                uxName.Text = "Name : " + basketData.Name;
                uxUserId.Text = "UserId : " + basketData.UserId;
                uxIsDefault.Text = "IsDefault : " + basketData.IsDefault;
                uxSubtotal.Text = "Subtotal : " + basketData.Subtotal;

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Default basket data does not
exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetDefault (shopper)

```
GetDefault(System.String)
```

Gets a default basket for a non-logged in user based upon the shopper's GUID string.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Shopper ID

Parameters

- `shopperId`. The ID of the non-logged in shopper.

Returns

Returns the default basket object of the shopper.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxShopperIdLabel" AssociatedControlID="uxShopperId"
CssClass="span-3 last" runat="server" Text="* Shopper Id:" />
        <ektronUI:TextField ID="uxShopperId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetDefault"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxuserIdres" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsDefault" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSubtotal" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            string shopperId = uxShopperId.Text;

            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            BasketData basketData = new BasketData();
            basketData = basketManager.GetDefault(shopperId);
            if (basketData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for visitor basket
with shopperId " + basketData.ShopperId.ToString() + " are below",
Message.DisplayModes.Success);
                uxuserIdres.Text = "Id : " + basketData.Id;
                uxName.Text = "Name : " + basketData.Name;
                uxIsDefault.Text = "IsDefault : " + basketData.IsDefault;
                uxSubtotal.Text = "Subtotal : " + basketData.Subtotal;

            }
            else
            {

                MessageUtilities.UpdateMessage(uxMessage, "Either visitor or basket data
with shopper ID " + shopperId + " does not exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetDefault (user)

```
GetDefault(System.Int64)
```

Gets the default basket of the specific user.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `userId`. The ID of the user.

Returns

Returns the default basket object of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last" runat="server" Text="* User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetDefault"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxuserIdres" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsDefault" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSubtotal" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

```
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long userId = long.Parse(uxUserId.Text);

            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            BasketData basketData = new BasketData();
            basketData = basketManager.GetDefault(userId);
            if (basketData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for user basket with
userId " + basketData.UserId.ToString() + " are below", Message.DisplayModes.Success);
                uxuserIdres.Text = "Id : " + basketData.Id;
                uxName.Text = "Name : " + basketData.Name;
                uxIsDefault.Text = "IsDefault : " + basketData.IsDefault;
                uxSubtotal.Text = "Subtotal : " + basketData.Subtotal;

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either user or basket data
with user ID " + userId + " does not exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a single basket by its ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket ID

Parameters

- `basketId`. ID of basket to be retrieved.

Returns

Returns the basket that matches the ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-3 last" runat="server" Text="* Basket Id:" />
        <ektronUI:TextField ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsDefault" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSubtotal" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long basketId = long.Parse(uxBasketId.Text);

            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            BasketData basketData = new BasketData();
            basketData = basketManager.GetItem(basketId);
            if (basketData != null)
            {

                MessageUtilities.UpdateMessage(uxMessage, "Details for basket  Id " +
basketData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxName.Text = "Name : " + basketData.Name;
                uxUserId.Text = "UserId : " + basketData.UserId;
                uxIsDefault.Text = "IsDefault : " + basketData.IsDefault;
                uxSubtotal.Text = "Subtotal : " + basketData.Subtotal;

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either basket data with ID "
+ basketId + " does not exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(System.Int64)
```

Retrieves a list of saved baskets of a user.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `userId`. ID of the user.

Returns

Returns a list of saved basket objects of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-5 last" runat="server" Text="* User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxBasketDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >No records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                 <tr>
                    <th>Id</th>
                    <th>Name</th>
                    <th>UserId</th>
                    <th>IsDefault</th>
                    <th>Subtotal</th>
                 </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
```

```
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsDefault")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Subtotal")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long userId = long.Parse(uxUserId.Text);

            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            List<BasketData> basketDataList = basketManager.GetList(userId);
            uxBasketDataListView.Visible = true;
            uxBasketDataListView.DataSource = basketDataList;
            uxBasketDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

RemoveCoupon

```
RemoveCoupon(System.Int64,System.Int64)
```

Removes the coupon from the basket.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket ID
- * Coupon ID

Parameters

- `basketId`. ID of the basket from which the coupon is removed.
- `couponId`. ID of the coupon to be removed from basket.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-3 last" runat="server" Text="* Basket Id:" />
        <ektronUI:TextField  ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="RemoveCoupon"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            long couponId = long.Parse(uxCouponId.Text);
            long basketId = long.Parse(uxBasketId.Text);

            basketManager.RemoveCoupon(basketId, couponId);

            MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
removed from basket "+ basketId , Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

SetDefault

```
SetDefault(System.Int64,System.Int64)
```

Sets the basket as default basket to the user.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket ID
- * User ID

Parameters

- `basketId`. ID of the basket from which the coupon is removed.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-5 last" runat="server" Text="* Basket Id:" />
        <ektronUI:TextField ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-5 last" runat="server" Text="* User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="SetDefault"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            long basketId = long.Parse(uxBasketId.Text);
            long userId = long.Parse(uxUserId.Text);
```

```
            BasketData basket = basketManager.GetItem(basketId);
            if (basket != null)
            {
                basketManager.SetDefault(basketId, userId);
                MessageUtilities.UpdateMessage(uxMessage, "Basket " + basketId + " is
maked as default basket to the user " + userId, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Basket Id is not valid. ",
Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.BasketData)
```

Updates the existing basket in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket ID
- Name
- Is Default

Parameters

- `basketData`. Basket object to be updated.

Returns

Returns the updated BasketData object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
```

```
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-3 last" runat="server" Text="* Basket Id:" />
        <ektronUI:TextField  ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxBasketNameLabel" AssociatedControlID="uxBasketName"
CssClass="span-3 last" runat="server" Text=" Name:" />
        <ektronUI:TextField  ID="uxBasketName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketDefaultLabel" AssociatedControlID="uxBasketDefault"
CssClass="span-3 last" runat="server" Text=" IsDefault:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxBasketDefault"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            Ektron.Cms.Framework.Commerce.BasketManager basketManager = new
Ektron.Cms.Framework.Commerce.BasketManager();
            bool isDefault = false;
            if (uxBasketDefault.Checked)
            {
                isDefault = true;
            }
            BasketData basketData = basketManager.GetItem(long.Parse(uxBasketId.Text));
            if (basketData != null)
```

```
            {
                basketData.Name = uxBasketName.Text;
                basketData.IsDefault = isDefault;
                basketManager.Update(basketData);
                MessageUtilities.UpdateMessage(uxMessage, "Basket is updated with Id " +
basketData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Basket id " + uxBasketId.Text
+ " does not exists", Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

BasketData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `BasketData()`

  ```
  public BasketData()
  ```

- `DateCreated`. Gets or sets the date created.

  ```
  public System.DateTime DateCreated { set; get; }
  ```

- `DateModified`. Gets or sets the date modified.

  ```
  public System.DateTime DateModified { set; get; }
  ```

- `HasRecurringItems`

  ```
  public bool HasRecurringItems { get; }
  ```

- `HasSubscriptions`

  ```
  public bool HasSubscriptionItems { get; }
  ```

- `HasTangibleItems`

  ```
  public bool HasTangibleItems { get; }
  ```

- `Id`. Gets or sets the Id of the Basket.

  ```
  public long Id { set; get; }
  ```

- `IsDefault`. Gets or sets value indicating whether or not this basket is the default basket for customers.

```
public bool IsDefault { set; get; }
```

- `Items`. Gets the collection items in the basket.

```
public Ektron.Cms.Commerce.BasketItemCollection Items { set; get; }
```

- `Name`. Gets or sets the name of the basket.

```
public string Name { set; get; }
```

- `ShippingAddressId`. Gets or sets the ID of the address this basket will be shipping to. This is used for calculating taxes, shipping, and so on.

```
public long ShippingAddressId { set; get; }
```

- `ShippingMethodId`. Gets or sets the ID of the shipping method to be used for the basket. This is used for calculating shipping costs.

```
public long ShippingMethodId { set; get; }
```

- `ShopperId`. Gets or sets the Shopper\visitor Id using the basket. This Id is used when customer is not logged in.

```
public string ShopperId { set; get; }
```

- `Subtotal`. Gets the basket subtotal.

```
public decimal Subtotal { get; }
```

- `UserId`. Gets or sets the Id of the user asscoiated with the basket.

```
public long UserId { set; get; }
```

CatalogEntryManager

9.00 and higher

The class manipulates the catalog entry data in CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.CatalogEntryManager
```

Constructors

- `CatalogEntryManager()`
- `CatalogEntryManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Submit* on page 408
- *Update* on page 410

Add

```
Add(Ektron.Cms.Commerce.EntryData)
```

Adds a new catalog entry into the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- * HTML
- * Catalog Folder ID
- * Currency ID
- * Tax Class ID
- * Product Type ID

Parameters

- `entryData`. The EntryData object to be added.

Returns

Returns added EntryData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-4
last" runat="server" Text="* Title:" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <asp:Label ID="uxHtmlLabel" AssociatedControlID="uxHtml" CssClass="span-4 last"
runat="server" Text="* Html:" />
        <asp:TextBox TextMode="MultiLine" Rows="4"  ID="uxHtml" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
```

```
CssClass="span-4 last" runat="server" Text="* Catalog FolderId:" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyIdLabel" AssociatedControlID="uxCurrencyId"
CssClass="span-4 last" runat="server" Text="* CurrencyId:" />
        <ektronUI:TextField ID="uxCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxClassIdLabel" AssociatedControlID="uxTaxClassId"
CssClass="span-4 last" runat="server" Text="* TaxClassId:" />
        <ektronUI:TextField ID="uxTaxClassId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxProductTypeIdLabel" AssociatedControlID="uxProductTypeId"
CssClass="span-4 last" runat="server" Text="* ProductTypeId:" />
        <ektronUI:TextField ID="uxProductTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
   <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long productTypeId = long.Parse(uxProductTypeId.Text);
            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            ProductTypeManager pTypeManager=new ProductTypeManager();
            Pricing pricingManager = new Pricing();

            Criteria<TierPriceProperty> tierCri = new Criteria<TierPriceProperty>();
```

```
            tierCri.AddFilter(TierPriceProperty.Id, CriteriaFilterOperator.GreaterThan,
0);
            List<CurrencyPricingData> addlist = new List<CurrencyPricingData>();
            addlist = pricingManager.GetCurrencyPricing(tierCri);
            PricingData priceData = new PricingData()
            {
                CurrencyPricelist = addlist
            };


            ProductData productData = new ProductData();
            ProductTypeData productTypeData = new Ektron.Cms.Commerce.ProductTypeData();
            productTypeData=pTypeManager.GetItem(productTypeId,true);
            productData.Title = uxTitle.Text;
            productData.Html = uxHtml.Text;
            productData.FolderId = long.Parse(uxFolderId.Text);
            productData.CurrencyId = long.Parse(uxCurrencyId.Text);
            productData.TaxClassId = long.Parse(uxTaxClassId.Text);
            productData.ProductType = productTypeData;
            productData.Pricing = priceData;
            catalogEntryManager.Add(productData);
            MessageUtilities.UpdateMessage(uxMessage, "Catalog entry Data is added with
Id " + productData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Cancel

```
Cancel(System.Int64)
```

Undo the last changes of catalog entry item.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `entryId`. ID of entry data item.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryIdId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Cancel"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxEntryId.Text);
            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            catalogEntryManager.Cancel(ID);
            MessageUtilities.UpdateMessage(uxMessage, "Catalog entry data with Id " +
uxEntryId.Text + " has been cancelled", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

CheckIn

```
CheckIn(System.Int64)
```

Check in the catalog entry item.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `entryId`. ID of entry data item to be checked in.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryIdId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="CheckIn"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxEntryId.Text);
            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            catalogEntryManager.CheckIn(ID);
            MessageUtilities.UpdateMessage(uxMessage, "Catalog entry data with Id " +
uxEntryId.Text + " has been checkin", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

CheckOut

```
CheckOut(System.Int64)
```

Check out the catalog entry item.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `entryId`. ID of entry data item to be checked out.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryIdId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="CheckOut"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxEntryId.Text);
            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            catalogEntryManager.CheckOut(ID);
            MessageUtilities.UpdateMessage(uxMessage, "Catalog entry data with Id " +
uxEntryId.Text + " has been checkout", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a catalog entry based on ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `entryId`. ID of entry data item to be deleted.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryIdId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
```

```
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxEntryId.Text);
            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            catalogEntryManager.Delete(ID);
            MessageUtilities.UpdateMessage(uxMessage, "Catalog entry data with Id " +
uxEntryId.Text + "is deleted", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DisableInventory

```
DisableInventory(System.Int64)
```

Disables the inventory option specific catalog entry item.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `entryId`. ID of the catalog entry item.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryIdId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="DisableInventory"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxEntryId.Text);
            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            catalogEntryManager.DisableInventory(ID);
            MessageUtilities.UpdateMessage(uxMessage, "Catalog entry inventory with Id "
+ uxEntryId.Text + " has been disabled", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

EnableInventory

```
EnableInventory(System.Int64)
```

Enables the inventory option specific catalog entry item.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `entryId`. ID of the catalog entry item.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryIdId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="EnableInventory"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxEntryId.Text);
            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            catalogEntryManager.EnableInventory(ID);
            MessageUtilities.UpdateMessage(uxMessage, "Catalog entry inventory with Id "
+ uxEntryId.Text + " has been enabled", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a single catalog entry in the current language based on the entry's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `entryId`. The ID of catalog entry to be retrieved.

Returns

Entry data object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
```

```
    <li class="clearfix">
        <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFolderId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCurrencyId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTaxClassId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsBuyable" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long entryId = long.Parse(uxEntryId.Text);

            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            ProductData productData = new ProductData();
            productData = (ProductData)catalogEntryManager.GetItem(entryId);
            if (productData != null)
            {

                MessageUtilities.UpdateMessage(uxMessage, "Details for catalog entry
data Id " + productData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxTitle.Text = "Title : " + productData.Title;
                uxFolderId.Text = "FolderId : " + productData.FolderId;
                uxCurrencyId.Text = "CurrencyId : " +
productData.Pricing.CurrencyPricelist[0].CurrencyId;
                uxTaxClassId.Text = "TaxClassId : " + productData.TaxClassId;
                uxIsBuyable.Text = "IsBuyable : " + productData.IsBuyable;


            }
```

```
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either catalog entry data
with ID " + entryId + " does not exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (catalog entries)

```
GetList(Ektron.Cms.Commerce.CatalogEntryCriteria)
```

Retrieves a list of EntryData objects based upon the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry Property
- * Object Value

Parameters

- `criteria`. The criteria for retrieving EntryData.

Returns

List of entry data object.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryPropertyLabel" AssociatedControlID="uxEntryProperty"
CssClass="span-4 last" runat="server" Text="* EntryProperty:" />
        <asp:DropDownList ID="uxEntryProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Title</asp:ListItem>
            <asp:ListItem>LanguageId</asp:ListItem>
```

```
            <asp:ListItem>CatalogId</asp:ListItem>
            <asp:ListItem>ProductTypeId</asp:ListItem>
            <asp:ListItem>CurrencyId</asp:ListItem>
            <asp:ListItem>TaxClassId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
 <asp:ListView ID="uxCatalogEntryDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria didn't match the records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                <th>Id</th>
                <th>Title</th>
                <th>FolderId</th>
                <th>CurrencyId</th>
                <th>TaxClassId</th>
                <th>IsBuyable</th>
                </tr>
            </thead>
            <tbody>
            <asp:PlaceHolder ID="aspItemPlaceholder" runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FolderId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("CurrencyId")%>
            </td>
                <td class="devsite-method">
```

```
                <%# Eval("TaxClassId")%>
            </td>
                <td class="devsite-method">
                <%# Eval("IsBuyable")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            CatalogEntryCriteria criteria = new CatalogEntryCriteria();
            if (uxObjectValue.Text == null || uxObjectValue.Text == "")
            {
                MessageUtilities.UpdateMessage(uxMessage, "Enter the criteria object
value ", Message.DisplayModes.Error);
                uxPageMultiView.SetActiveView(uxViewMessage);
            }
            else
            {
                if (uxEntryProperty.SelectedItem.Text == "Id")
                {
                    Objectvalue = long.Parse(uxObjectValue.Text);
                    criteria.AddFilter(EntryProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
                }
                else if (uxEntryProperty.SelectedItem.Text == "Title")
                {
                    Objectvalue = uxObjectValue.Text;
                    criteria.AddFilter(EntryProperty.Title,
CriteriaFilterOperator.EqualTo, Objectvalue);
                }
                else if (uxEntryProperty.SelectedItem.Text == "CatalogId")
                {
                    Objectvalue = uxObjectValue.Text;
                    criteria.AddFilter(EntryProperty.CatalogId,
CriteriaFilterOperator.EqualTo, Objectvalue);
```

```
                }
                else if (uxEntryProperty.SelectedItem.Text == "LanguageId")
                {
                    Objectvalue = uxObjectValue.Text;
                    criteria.AddFilter(EntryProperty.LanguageId,
CriteriaFilterOperator.EqualTo, Objectvalue);
                }
                else if (uxEntryProperty.SelectedItem.Text == "ProductTypeId")
                {
                    Objectvalue = uxObjectValue.Text;
                    criteria.AddFilter(EntryProperty.ProductTypeId,
CriteriaFilterOperator.EqualTo, Objectvalue);
                }
                else if (uxEntryProperty.SelectedItem.Text == "CurrencyId")
                {
                    Objectvalue = uxObjectValue.Text;
                    criteria.AddFilter(EntryProperty.CurrencyId,
CriteriaFilterOperator.EqualTo, Objectvalue);
                }
                else
                {
                    Objectvalue = uxObjectValue.Text;
                    criteria.AddFilter(EntryProperty.TaxClassId,
CriteriaFilterOperator.EqualTo, Objectvalue);
                }

                List<EntryData> entryDataList = catalogEntryManager.GetList(criteria);
                uxCatalogEntryDataListView.Visible = true;
                uxCatalogEntryDataListView.DataSource = entryDataList;
                uxCatalogEntryDataListView.DataBind();

                uxPageMultiView.SetActiveView(uxViewMessage);
            }
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (entry attribute)

```
GetList(Ektron.Cms.Commerce.EntryAttributeCriteria)
```

Retrieves a list of EntryData objects based upon the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry Attribute Property
- * Object Value
- * Attribute Value

Parameters

- `criteria`. EntryAttributeCriteria.

Returns

Returns list of entry data object.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryPropertyLabel" AssociatedControlID="uxEntryProperty"
CssClass="span-4 last" runat="server" Text="* EntryAttributeProperty:" />
        <asp:DropDownList ID="uxEntryProperty" runat="server">
            <asp:ListItem>AttributeId</asp:ListItem>
            <asp:ListItem>AttributeName</asp:ListItem>
        </asp:DropDownList>
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAttributeValueLabel"
AssociatedControlID="uxAttributeValue" CssClass="span-4 last" runat="server" Text="*
AttributeValue:" />
        <ektronUI:TextField ID="uxAttributeValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
 <asp:ListView ID="uxCatalogEntryDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria didn't match the records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
```

```
            <thead>
                <tr>
                <th>Id</th>
                <th>Title</th>
                <th>FolderId</th>
                <th>CurrencyId</th>
                <th>TaxClassId</th>
                <th>IsBuyable</th>
                </tr>
            </thead>
            <tbody>
            <asp:PlaceHolder ID="aspItemPlaceholder" runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FolderId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("CurrencyId")%>
            </td>
                <td class="devsite-method">
                <%# Eval("TaxClassId")%>
            </td>
                <td class="devsite-method">
                <%# Eval("IsBuyable")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
```

```
        {
            object Objectvalue;

            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            EntryAttributeCriteria criteria = new EntryAttributeCriteria();

            if (uxObjectValue.Text == null || uxObjectValue.Text=="")
            {
                MessageUtilities.UpdateMessage(uxMessage, "Enter the criteria object
value ", Message.DisplayModes.Error);

            }
            else if (uxAttributeValue.Text == "")
            {
                MessageUtilities.UpdateMessage(uxMessage, "Enter the attribute value",
Message.DisplayModes.Error);

            }
            else
            {
                if (uxEntryProperty.SelectedItem.Text == "AttributeId")
                {
                    Objectvalue = long.Parse(uxObjectValue.Text);
                    criteria.AddFilter(EntryAttributeCriteriaProperty.AttributeId,
CriteriaFilterOperator.EqualTo, Objectvalue);
                }
                else
                {
                    Objectvalue = uxObjectValue.Text;
                    criteria.AddFilter(EntryAttributeCriteriaProperty.AttributeName,
CriteriaFilterOperator.EqualTo, Objectvalue);
                }
                criteria.AddFilter(EntryAttributeCriteriaProperty.AttributeValue,
CriteriaFilterOperator.EqualTo, uxAttributeValue.Text);
                List<EntryData> entryDataList = catalogEntryManager.GetList(criteria);
                uxCatalogEntryDataListView.Visible = true;
                uxCatalogEntryDataListView.DataSource = entryDataList;
                uxCatalogEntryDataListView.DataBind();

            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Restore

```
Restore(System.Int64,System.Int64)
```

Restores a previous version of a catalog entry based on its ID and the version ID to restore.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Version ID

Parameters

- `entryId`. The ID of the catalog entry to restore.
- `versionId`. Version ID to restore.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxVersionIdLabel" AssociatedControlID="uxVersionId"
CssClass="span-3 last" runat="server" Text="* VersionId:" />
        <ektronUI:TextField ID="uxVersionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFolderId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
```

```
        <asp:Literal ID="uxCurrencyId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTaxClassId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsBuyable" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long entryId = long.Parse(uxEntryId.Text);
            long versionId = long.Parse(uxVersionId.Text);
            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            catalogEntryManager.Restore(entryId, versionId);
            MessageUtilities.UpdateMessage(uxMessage, "Catalog entry data has been
restored", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Submit

```
Submit(System.Int64)
```

Submits the catalog entry item for publish.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `entryId`. ID of catalog entry item.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryIdId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Submit"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxEntryId.Text);
            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            catalogEntryManager.Submit(ID);
            MessageUtilities.UpdateMessage(uxMessage, "Catalog entry data with Id " +
uxEntryId.Text + " has been submitted", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.EntryData)
```

Updates an existing catalog entry data.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- * HTML
- * Currency ID
- * Tax Class ID

Parameters

- `entryData`. The EntryData object to be updated.

Returns

Returns updated EntryData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryId"
CssClass="span-4 last" runat="server" Text=" Id:" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-4
last" runat="server" Text="* Title:" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <asp:Label ID="uxHtmlLabel" AssociatedControlID="uxHtml" CssClass="span-4 last"
runat="server" Text="* Html:" />
        <asp:TextBox TextMode="MultiLine" Rows="4"  ID="uxHtml" CssClass="span-6"
```

```
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyIdLabel" AssociatedControlID="uxCurrencyId"
CssClass="span-4 last" runat="server" Text="* CurrencyId:" />
        <ektronUI:TextField ID="uxCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxClassIdLabel" AssociatedControlID="uxTaxClassId"
CssClass="span-4 last" runat="server" Text="* TaxClassId:" />
        <ektronUI:TextField ID="uxTaxClassId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long entryId = long.Parse(uxEntryId.Text);
            CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
            ProductData productData = new ProductData();
            productData = (ProductData)catalogEntryManager.GetItem(entryId);

            if (entryId > 0 && productData != null)
            {
                productData.Title = uxTitle.Text;
                productData.Html = uxHtml.Text;
                productData.CurrencyId = long.Parse(uxCurrencyId.Text);
                productData.TaxClassId = long.Parse(uxTaxClassId.Text);
```

```
                if (productData.Status != "O")
                {
                    catalogEntryManager.CheckOut(productData.Id);
                }
                catalogEntryManager.Update(productData);
                MessageUtilities.UpdateMessage(uxMessage, "Catalog entry data is updated
with Id " + uxEntryId.Text, Message.DisplayModes.Success);

            }
            else
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid catalog
entry ID.", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

CatalogEntryCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `CatalogEntryCriteria()`

  ```
  public CatalogEntryCriteria()
  ```

- `CatalogEntryCriteria(Ektron.Cms.Commerce.EntryProperty, EkEnumeration.OrderByDirection)`

  ```
  public CatalogEntryCriteria(Ektron.Cms.Commerce.EntryProperty
    entryProperty, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `EntryProperty`are:

  - AverageRating
  - CatalogId
  - CollItemsDisplayOrder
  - ContentStatus
  - CurrencyId
  - EndDate

- EntryType
- GoLive
- Html
- Id
- IsArchived
- IsBuyable
- IsPublished
- IsSearchable
- LanguageId
- LastEditDate
- LastEditorFirstName
- LasteditorLastName
- ListPrice
- Media
- NumberRated
- ProductTypeId
- ReturnEntryPricebookprice
- SalePrice
- Sku
- Status
- Summary
- TaxClassId
- TaxItemDisplayOrder
- Title
- ViewCount

EntryAttributeCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `AddAttributeFilter(string, Ektron.Cms.Common.CriteriaFilterOperator, object)`

```
public Ektron.Cms.Common.CriteriaFilterGroup<EntryAttributeCriteriaProperty>
  AddAttributeFilter(string attributeName,
    Ektron.Cms.Common.CriteriaFilterOperator filterOperator,
    object value)
```

- `AddAttributeFilter(long, Ektron.Cms.Common.CriteriaFilterOperator, object)`

```
public
  Ektron.Cms.Common.CriteriaFilterGroup<EntryAttributeCriteriaProperty>
  AddAttributeFilter(long attributeId,
    Ektron.Cms.Common.CriteriaFilterOperator filterOperator,
    object value)
```

- `AddFilter(Ektron.Cms.Commerce.EntryAttributeCriteriaProperty, Ektron.Cms.Common.CriteriaFilterOperator, object)`

```
public override
  Ektron.Cms.Common.CriteriaFilter<EntryAttributeCriteriaProperty>
  AddFilter(Ektron.Cms.Commerce.EntryAttributeCriteriaProperty field,
    Ektron.Cms.Common.CriteriaFilterOperator filterOperator,
    object value)
```

- `EntryAttributeCriteria()`

```
public EntryAttributeCriteria()
```

- `EntryAttributeCriteria (Ektron.Cms.Commerce.EntryAttributeCriteriaProperty, EkEnumeration.OrderByDirection)`

```
public
  EntryAttributeCriteria(Ektron.Cms.Commerce.EntryAttributeCriteriaProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `EntryAttributeCriteriaProperty` are:

- AttributeDataType
- AttributeDisplayOrder
- AttributeId
- AttributeName
- AttributeValue
- CatalogId
- CurrencyId
- EndDate
- EntryId
- EntryType
- GoLive
- IsArchived
- IsBuyable
- IsPublished
- IsSearchable
- LanguageId
- LastEditDate
- LastEditorFirstName
- LasteditorLastName
- ListPrice
- PricebookId

- PricebookPrice
- ProductTypeId
- SalePrice
- Sku
- TaxClassId
- Title

- `FindAttributeValueFilter(string)`

```
public Ektron.Cms.Common.CriteriaFilter<EntryAttributeCriteriaProperty>
  FindAttributeValueFilter(string attributeName)
```

- `FindAttributeValueFilter(long)`

```
public Ektron.Cms.Common.CriteriaFilter<EntryAttributeCriteriaProperty>
  FindAttributeValueFilter(long attributeId)
```

EntryData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `AnalyticsInfo`

```
public AnalyticsData AnalyticsInfo { set; get; }
```

- `Attributes`. Gets or sets the attributes of the Catalog Entry.

```
public System.Collections.Generic.List<EntryAttributeData>
  Attributes { set; get; }
```

- `Comment`

```
public string Comment { set; get; }
```

- `ContentStatus`. The status of the content information.

```
public string ContentStatus { set; get; }
```

- `CurrencyId`. Gets or sets the Id of the currency used for the current Sale and List prices.

```
public long CurrencyId { set; get; }
```

- `DateModified`

```
public System.DateTime DateModified { set; get; }
```

- `Dimensions`. Gets or sets the dimensions of the product. Used for shipping.

```
public Ektron.Cms.Commerce.Dimensions Dimensions { set; get; }
```

- `DisableInventoryManagement`. Gets or sets the disable inventory management flag. If true, this product does not track inventory and no inventory management is performed.

```
public bool DisableInventoryManagement { set; get; }
```

- `EndDate`. Get or sets the enddate of the EntryData object. Returns the datetime of the entry enddate.

```
public System.DateTime EndDate { set; get; }
```

- `EndDateAction`

```
public int EndDateAction { set; get; }
```

- `EntryData()`

```
public EntryData()
```

- `EntryData(EkEnumeration.CatalogEntryType)`

```
public EntryData(EkEnumeration.CatalogEntryType entryType)
```

- `EntryData(int, EkEnumeration.CatalogEntryType)`

```
public
  EntryData(int languageId, EkEnumeration.CatalogEntryType
  entryType)
```

- `EntryType`. The type of entry. Returns an integer of the entry quantity multiple.

```
public virtual EkEnumeration.CatalogEntryType EntryType
  { set; get; }
```

- `FolderId`. Gets or sets the folder ID of the EntryData object. Returns the ID of the entry folder.

```
public long FolderId { set; get; }
```

- `GoLive`. Get or sets the golive of the EntryData object. Returns the datetime of the entry golive.

```
public System.DateTime GoLive { set; get; }
```

- `Html`. Get or sets the HTML or XML of the EntryData object. Returns the string of the entry HTML.

```
public string Html { set; get; }
```

- `Id`. Gets or sets the ID of the EntryData object. Returns the ID of the entry title.

```
public long Id { set; get; }
```

- `Image`

```
public string Image { set; get; }
```

- `Imagethumbnail`

```
public string ImageThumbnail { set; get; }
```

- `IsArchived`. Get or sets the Archive flag of the EntryData object.

```
public bool IsArchived { set; get; }
```

- `IsAvailable`

```
public bool IsAvailable { set; get; }
```

- `IsBuyable`. Get or sets whether the entry is buyable. Returns a Boolean of the entry flag Buyable.

```
public bool IsBuyable { set; get; }
```

- `IsMarkedForDeletion`. Get or sets the MarkForDeletion flag on the EntryData object. Returns a Boolean of the entry flag MarkForDeletion.

```
public bool IsMarkedForDeletion { set; get; }
```

- `IsPublished`.

```
public bool IsPublished { set; get; }
```

- `IsSearchable`. Gets or sets whether the entry is searchable for the object. True = searchable. Returns a boolean based on whether the entry is searchable. * True = entry is searchable. * False = entry is not searchable.

```
public bool IsSearchable { set; get; }
```

- `IsTangible`. Indicates if this product is a tangible good. Returns a Boolean value indicating if this product is a tangible good.

```
public bool IsTangible { get; }
```

- `LanguageId`. Gets or sets the language of the EntryData object.

```
public int LanguageId { set; get; }
```

- `LastEditorFirstName`.

```
public string LastEditorFirstName { set; get; }
```

- `LastEditorId`

```
public long LastEditorId { set; get; }
```

- `LastEditorLastName`

```
public string LastEditorLastName { set; get; }
```

- `ListPrice`. Gets the current List price from the Pricing object. To set an item's price, Use the Pricing property. Create a TierPrice with a quantity of 1. Use the PricingData(CurrencyId, SalePrice, ListPrice) constructor.

```
public decimal ListPrice { get; }
```

- `Media`

```
public Ektron.Cms.MediaGalleryData Media { set; get; }
```

- `Metadata`

```
public System.Collections.Generic.List<ContentMetaData>
  Metadata { set; get; }
```

- `NumberSold`. Gets or sets the Number Sold for this entry. Returns a content item's Quicklink information.

```
public long NumberSold { set; get; }
```

- `Pricing`. The entry pricing, including currencies and quantity. Returns the pricing data of the catalog entry.

```
public Ektron.Cms.Commerce.PricingData Pricing { set; get; }
```

- `ProductType`

```
public Ektron.Cms.Commerce.ProductTypeData ProductType
  { set; get; }
```

- `QuantityMultiple`. The catalog has these number of items in it. Returns the integer of the entry quantity multiple.

```
public int QuantityMultiple { set; get; }
```

- `Quicklink`. Gets or sets the entry's Quicklink. Returns a content item's Quicklink information.

```
public string Quicklink { set; get; }
```

- `SalePrice`. Gets the current Sale price from the Pricing object. To set an item's sale price, Use the Pricing property. Create a TierPrice with a quantity of 1. Use

the PricingData(CurrencyId, SalePrice, ListPrice) constructor.

```
public decimal SalePrice { get; }
```

- `Sku`. Get or sets the sku of the EntryData object. Returns the string of the entry SKU.

```
public string Sku { set; get; }
```

- `Status`. The status of the entry information.

```
public string Status { set; get; }
```

- `StatusLanguage`. The language that the entry is checked out to, if any.

```
public int StatusLanguage { set; get; }
```

- `Summary`. Get or sets the summary of the EntryData object. Returns the string of the entry summary.

```
public string Summary { set; get; }
```

- `TaxClassId`

```
public long TaxClassId { set; get; }
```

- `TemplateId`

```
public long TemplateId { set; get; }
```

- `Title`. Get or sets the title of the EntryData object. Returns the string of the entry URL.

```
public string Title { set; get; }
```

- `Url`. Get or sets the url of the EntryData object. Returns the string of the entry URL.

```
public string Url { set; get; }
```

- `VersionId`. The version Id of the catalog entry. 0 is the current version.

```
public long VersionId { set; get; }
```

- `Weight`. Gets or sets the weight of the product. Used for shipping.

```
public Ektron.Cms.Commerce.Weight Weight { set; get; }
```

CountryManager

9.00 and higher

The class manipulates the country data in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.CountryManager
```

Constructors

- `CountryManager()`
- `CountryManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Commerce.CountryData)
```

Adds a new country to Ektron CMS based on the CountryData object. The Country.Id property is populated with the new Country's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Country ID
- * Country Name
- * Long ISO Code
- * Short ISO Code

Parameters

- `countryData`. The CountryData object to add.

Returns

Returns newly created CountryData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIDLabel" AssociatedControlID="uxCountryID"
CssClass="span-4 last"
            runat="server" Text="* Country ID :" />
        <ektronUI:TextField ID="uxCountryID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryNameLabel" AssociatedControlID="uxCountryName"
CssClass="span-4 last"
            runat="server" Text="* Country Name :" />
        <ektronUI:TextField ID="uxCountryName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLongIsoCodeLabel" AssociatedControlID="uxLongIsoCode"
CssClass="span-4 last"
            runat="server" Text="* LongIsoCode :" />
        <ektronUI:TextField ID="uxLongIsoCode" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShortIsoCodelabel" AssociatedControlID="uxShortIsoCode"
CssClass="span-4 last"
            runat="server" Text="* ShortIsoCode :" />
```

```
        <ektronUI:TextField ID="uxShortIsoCode" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            if (Convert.ToInt32(uxCountryID.Text) > 0 && uxCountryName.Text.ToString()
!= "" && uxLongIsoCode.Text.ToString() != "" && uxShortIsoCode.Text.ToString() != "")
            {
                CountryManager countryManager = new CountryManager();
                CountryData countryData = new CountryData();

                countryData.Id = Convert.ToInt32(uxCountryID.Text);
                countryData.Name = uxCountryName.Text.ToString();
                countryData.Enabled = true;
                countryData.LongIsoCode = uxLongIsoCode.Text.ToString();
                countryData.ShortIsoCode = uxShortIsoCode.Text.ToString();

                countryManager.Add(countryData);
                MessageUtilities.UpdateMessage(uxMessage, "Country added with Id: " +
countryData.Id.ToString(), Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
```

```
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

CanDelete

```
CanDelete(System.Int32)
```

Checks to see if a country can be deleted. A country cannot be deleted if a region or address uses the country.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Country ID

Parameters

- `countryId`. The ID of the country to delete.

Returns

Returns true if country can safely be deleted.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIDLabel" AssociatedControlID="uxCountryID"
CssClass="span-4 last"
            runat="server" Text="* Country ID :" />
        <ektronUI:TextField ID="uxCountryID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Check"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Microsoft.Practices.EnterpriseLibrary.Validation;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            int  countryId = int.Parse(uxCountryID.Text);
            if (countryId > 0)
            {
                CountryManager countryManager = new CountryManager();
                CountryData countryData = countryManager.GetItem(countryId);
                if (countryData != null)
                {
                    bool canDelete = countryManager.CanDelete(countryId);
                    if (canDelete == true)
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "The Country with Id "
+ countryId.ToString() + " is deletable", Message.DisplayModes.Success);
                    }
                    else
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "The Country with Id "
+ countryId.ToString() + " is not deletable", Message.DisplayModes.Error);
                    }
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Country with Id " +
countryId.ToString() + " could not be found", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Country
Id", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int32)
```

Deletes a country from the Ektron CMS based on its ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Country ID

Parameters

- `countryId`. The ID of the country to be deleted.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIDLabel" AssociatedControlID="uxCountryID"
CssClass="span-4 last"
            runat="server" Text="* Country ID :" />
        <ektronUI:TextField ID="uxCountryID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
```

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            int countryId = int.Parse(uxCountryID.Text);
            if (countryId > 0)
            {
                CountryManager countryManager = new CountryManager();
                CountryData countryData = countryManager.GetItem(countryId);
                if (countryData != null)
                {
                    countryManager.Delete(countryId);
                    MessageUtilities.UpdateMessage(uxMessage, "Country Deleted for Id: "
+ countryId.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Country with Id " +
countryId.ToString() + " could not be found", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Country
Id", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int32)
```

Retrieves a country by its ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Country ID

Parameters

- `countryId`. The ID of the country to retrieve.

Returns

Returns a CountryData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIDLabel" AssociatedControlID="uxCountryID"
CssClass="span-4 last"
            runat="server" Text="* Country ID :" />
        <ektronUI:TextField ID="uxCountryID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCountryName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLongIsoCode" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxShortIsoCode" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
```

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            int countryId = int.Parse(uxCountryID.Text);
            if (countryId > 0)
            {
                CountryManager countryManager = new CountryManager();
                CountryData countryData = countryManager.GetItem(countryId);
                if (countryData != null)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Country Details for Id: "
+ countryId.ToString() + " are below", Message.DisplayModes.Success);
                    uxCountryName.Text = "Country Name: " + countryData.Name;
                    uxLongIsoCode.Text = "LongIsoCode: " + countryData.LongIsoCode;
                    uxShortIsoCode.Text = "ShortIsoCode: " + countryData.ShortIsoCode;
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
Country Id", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Country
Id", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.CountryCriteria)
```

Retrieves a list of countries based on filters set in the CountryCriteria property.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Country Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter the list of countries being retrieved.

Returns

A list of countries meeting criteria filters.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryPropertyLabel"
AssociatedControlID="uxCountryProperty"
            CssClass="span-4 last" runat="server" Text="* CountryProperty :" />
        <asp:DropDownList ID="uxCountryProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
            <asp:ListItem>LongIsoCode</asp:ListItem>
        </asp:DropDownList>
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last"
            runat="server" Text="* ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCountrylistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
```

```
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Name
                    </th>
                    <th>
                        LongIsoCode
                    </th>
                    <th>
                        ShortIsoCode
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("LongIsoCode")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ShortIsoCode")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CountryManager countryManager = new CountryManager();
            object objectValue = uxObjectValue.Text;
            string countryProperty = uxCountryProperty.SelectedItem.Text;

            CountryCriteria criteria = new CountryCriteria();
            criteria.OrderByField = CountryProperty.Id;
            criteria.OrderByDirection = EkEnumeration.OrderByDirection.Ascending;

            if (countryProperty == "Id")
            {
                criteria.AddFilter(CountryProperty.Id, CriteriaFilterOperator.EqualTo,
objectValue);
            }
            else if (countryProperty == "Name")
            {
                criteria.AddFilter(CountryProperty.Name, CriteriaFilterOperator.EqualTo,
objectValue);
            }
            else
            {
                criteria.AddFilter(CountryProperty.LongIsoCode,
CriteriaFilterOperator.EqualTo, objectValue);
            }

            List<CountryData> countryDataList = countryManager.GetList(criteria);
            uxCountrylistView.Visible = true;
            uxCountrylistView.DataSource = countryDataList;
            uxCountrylistView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.CountryData)
```

Updates an existing country in Ektron CMS based on the CountryData object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Country ID
- Country Name
- Long ISO Code
- Short ISO Code

Parameters

- `countryData`. The CountryData object to save.

Returns

Returns updated CountryData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIDLabel" AssociatedControlID="uxCountryID"
CssClass="span-4 last"
            runat="server" Text="* Country ID :" />
        <ektronUI:TextField ID="uxCountryID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryNameLabel" AssociatedControlID="uxCountryName"
CssClass="span-4 last"
            runat="server" Text=" Country Name :" />
        <ektronUI:TextField ID="uxCountryName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLongIsoCodeLabel" AssociatedControlID="uxLongIsoCode"
CssClass="span-4 last"
            runat="server" Text=" LongIsoCode :" />
        <ektronUI:TextField ID="uxLongIsoCode" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShortIsoCodelabel" AssociatedControlID="uxShortIsoCode"
CssClass="span-4 last"
            runat="server" Text=" ShortIsoCode :" />
        <ektronUI:TextField ID="uxShortIsoCode" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            int countryId = int.Parse(uxCountryID.Text);
            if (countryId > 0)
            {
                CountryManager countryManager = new CountryManager();
                CountryData countryData = countryManager.GetItem(countryId);
                countryData.Name = uxCountryName.Text == "" ? countryData.Name :
uxCountryName.Text;
                countryData.LongIsoCode = uxLongIsoCode.Text == "" ?
countryData.LongIsoCode : uxLongIsoCode.Text;
                countryData.ShortIsoCode = uxShortIsoCode.Text == "" ?
countryData.ShortIsoCode : uxShortIsoCode.Text;
                countryManager.Update(countryData);
                MessageUtilities.UpdateMessage(uxMessage, "Country updated for Id: " +
countryId.ToString() , Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Country
Id", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Data Classes

CountryCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `CountryCriteria()`

  ```
  public CountryCriteria()
  ```

- `CountryCriteria(Ektron.Cms.Commerce.CountryProperty, EkEnumeration.OrderByDirection)`

  ```
  public
    CountryCriteria(Ektron.Cms.Commerce.CountryProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `CountryProperty` are:

  - Id
  - IsEnabled
  - LongIsoCode
  - Name
  - ShortIsoCode

CountryData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `CountryData()`

  ```
  public CountryData()
  ```

- `CountryData(int, string, string, string, bool)`

  ```
  public CountryData(int id, string name, string shortIsoCode,
    string longIsoCode, bool enabled)
  ```

- `Enabled`. Gets or sets whether or not the country is enabled for use.

  ```
  public bool Enabled { set; get; }
  ```

- `Id`. Gets or sets the ID of the country. The ID is the ISO numeric code of the country. This property is required when creating an country data object.

  ```
  public int Id { set; get; }
  ```

- `LongIsoCode`. Gets or sets the 3 letter ISO code of the country. This property is required when creating an country data object.

```
public string LongIsoCode { set; get; }
```

- `Name`. Gets or sets the name of the country. This property is required when creating an country data object.

```
public string Name { set; get; }
```

- `ShortIsoCode`. Gets or sets the 2 letter ISO code of the country. This property is required when creating an country data object.

```
public string ShortIsoCode { set; get; }
```

- `ToString`. Returns Name of Country.

```
public override string ToString()
```

- `Validate`. Validates the CountryData object is valid for saving. Returns a ValidationResult object with any validation errors.

```
public ValidationResults Validate()
```

CouponManager

9.00 and higher

The class manipulates the coupon data in the CMS. It can be used for add, edit, and get the coupon data object.

Namespace

```
Ektron.Cms.Framework.Commerce.CouponManager
```

Constructors

- `CouponManager()`
- `CouponManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Commerce.CouponData)
```

Adds the new coupon data object into the CMS.The CouponData.Id property is populated with the new Coupon's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon Code
- Coupon Type
- Is Active

Parameters

- `couponData`. Coupon data object to be added.

Returns

CouponData object with new coupon ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponCodeLabel" AssociatedControlID="uxCouponCode"
CssClass="span-3 last" runat="server" Text="* Coupon Code:" />
        <ektronUI:TextField  ID="uxCouponCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponTypeLabel" AssociatedControlID="uxCouponType"
CssClass="span-3 last" runat="server" Text=" Coupon Type:" />
         <asp:DropDownList ID="uxCouponType" runat="server">
            <asp:ListItem Text="BasketLevel" Value="0" >Group Id</asp:ListItem>
```

```
            <asp:ListItem Text="MostExpensiveItem" Value="1" >Group Id</asp:ListItem>
            <asp:ListItem Text="LeastExpensiveItem" Value="2" >Group Id</asp:ListItem>
            <asp:ListItem Text="AllItems" Value="3" >Group Id</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponActiveLabel" AssociatedControlID="uxCouponActive"
CssClass="span-3 last" runat="server" Text=" IsActive:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxCouponActive"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            bool isActive = false;
            if (uxCouponActive.Checked)
            {
                isActive = true;
            }
            CouponData couponData = new CouponData()
            {
                Code = uxCouponCode.Text,
                CouponType = (Ektron.Cms.Common.EkEnumeration.CouponType)int.Parse
(uxCouponType.SelectedItem.Value),
                IsActive = isActive,

            };

            CouponData newCouponData = couponManager.Add(couponData);
            if (newCouponData.Id > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Coupon is added with Id " +
couponData.Id , Message.DisplayModes.Success);
            }
            else
```

```
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon adding process is
failed", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

AddCouponToObject

```
AddCouponToObject(Ektron.Cms.Commerce.CouponEntryData)
```

Adds new CouponEntry data object into the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID
- * Object ID
- Object Type
- Is Included

Parameters

- `couponEntryData`. CouponEntryData object to be added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
      <li class="clearfix">
        <ektronUI:Label ID="uxObjectIdLabel" AssociatedControlID="uxObjectId"
CssClass="span-3 last" runat="server" Text="* Object Id:" />
        <ektronUI:TextField  ID="uxObjectId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponTypeLabel" AssociatedControlID="uxCouponType"
CssClass="span-3 last" runat="server" Text=" Coupon Type:" />
         <asp:DropDownList ID="uxCouponType" runat="server">
            <asp:ListItem Value="1">Folder</asp:ListItem>
            <asp:ListItem Value="20">TaxonomyNode</asp:ListItem>
            <asp:ListItem Value="21">CatalogEntry</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIncludedLabel" AssociatedControlID="uxCouponActive"
CssClass="span-3 last" runat="server" Text=" Is Active:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxCouponActive"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Add
Coupon Application"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            bool isInclude = false;
            if (uxCouponInclude.Checked)
            {
                isInclude = true;
            }
            CouponEntryData couonEntryData = new CouponEntryData()
            {
                CouponId = long.Parse(uxCouponId.Text),
                DataState = Ektron.Cms.Common.EkEnumeration.DataState.Added,
                ObjectType = (Ektron.Cms.Common.EkEnumeration.CMSObjectTypes)int.Parse
```

```
(uxObjectType.SelectedValue),
                ObjectId = long.Parse(uxObjectId.Text),
                IsIncluded = isInclude
            };
            couponManager.AddCouponToObject(couonEntryData);
            MessageUtilities.UpdateMessage(uxMessage, "Coupon "+ couonEntryData.CouponId
+ " is added to object " + couonEntryData.ObjectId , Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Deactivate

```
Deactivate(System.Int64)
```

Deactivate the existing coupon.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID

Parameters

- `couponId`. ID of the coupon.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Deactivate"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            long couponId = long.Parse(uxCouponId.Text);

            couponManager.Deactivate(couponId);
            MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
deactivated", Message.DisplayModes.Success );

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Delete

```
Delete(System.Int64)
```

Deletes an existing coupon from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID

Parameters

- `couponId`. ID of the coupon to be deleted.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            long couponId = long.Parse(uxCouponId.Text);

            couponManager.Delete(couponId);
            MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
deleted", Message.DisplayModes.Success );

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeleteCouponApplications

```
DeleteCouponApplications
(System.Int64,System.Collections.Generic.IEnumerable,Ektron.Cms.Common.EkEnumeration.CMS
ObjectTypes)
```

Deletes the existing Coupon Applications from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID
- * Object IDs
- Object Type

Parameters

- `couponId`. ID of the coupon.
- `exclusionIdList`. List of IDs not included for delete.
- `objectType`. Object type of CMSObjectTypes. Only Folder, Taxonomy and CatalogEntry object types are allowed.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
      <li class="clearfix">
        <ektronUI:Label ID="uxObjectIdLabel" AssociatedControlID="uxObjectId"
CssClass="span-3 last" runat="server" Text="* Object Id:" />
        <ektronUI:TextField  ID="uxObjectId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponTypeLabel" AssociatedControlID="uxCouponType"
CssClass="span-3 last" runat="server" Text=" Coupon Type:" />
         <asp:DropDownList ID="uxCouponType" runat="server">
            <asp:ListItem Value="1">Folder</asp:ListItem>
```

```
            <asp:ListItem Value="20">TaxonomyNode</asp:ListItem>
            <asp:ListItem Value="21">CatalogEntry</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="DeleteCoupon Application"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            long couponId=0;
             long.TryParse(uxCouponId.Text,out couponId);
            List<string> list = uxObjectIds.Text.Split(',').ToList();
            List<long> listObjectIds = list.ConvertAll<long>(delegate(string x) { return
long.Parse(x); });
            Ektron.Cms.Common.EkEnumeration.CMSObjectTypes objectType =
(Ektron.Cms.Common.EkEnumeration.CMSObjectTypes)int.Parse(uxObjectType.SelectedValue);

            couponManager.DeleteCouponApplications(couponId, listObjectIds, objectType);
            MessageUtilities.UpdateMessage(uxMessage, "Coupon application is deleted",
Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetCatalogList

```
GetCatalogList(Ektron.Cms.Commerce.CouponEntryCriteria)
```

Retrieves the list of coupon entry item based on the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Coupon Entry Property
- * Object Value

Parameters

- `criteria`. Criteria used to retrieve the coupon entry.

Returns

Returns the list of CouponEntryData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCouponEntryPropertyLabel"
AssociatedControlID="uxCouponEntryProperty" CssClass="span-4 last" runat="server"
Text="CouponEntryProperty:" />
        <asp:DropDownList ID="uxCouponEntryProperty" runat="server">
            <asp:ListItem>Coupon Id</asp:ListItem>
            <asp:ListItem>Object ID</asp:ListItem>
            <asp:ListItem>Title </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCouponEntryListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Coupon Id
                    </th>
                    <th>
                        Object Id
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
                        Path
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("CouponId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Path")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            CouponEntryCriteria criteria = new CouponEntryCriteria();

            if (uxCouponEntryProperty.SelectedItem.Text == "Coupon Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(CouponEntryProperty.CouponId ,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo,Objectvalue );
            }
            else if (uxCouponEntryProperty.SelectedItem.Text == "Object Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(CouponEntryProperty.ObjectId  ,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo,Objectvalue );
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CouponEntryProperty.Title,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo,Objectvalue );
            }

            CouponManager couponManger = new CouponManager();
            List<CouponEntryData> couponEntryList = couponManger.GetCatalogList
(criteria);

            uxCouponEntryListView.Visible = true;
            uxCouponEntryListView.DataSource = couponEntryList;
            uxCouponEntryListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetCouponId

```
GetCouponId(System.String)
```

Retrieve the coupon ID by coupon code.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon Code

Parameters

- `couponCode`. Code of the coupon.

Returns

Returns the coupon ID.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxCouponCodeLabel" AssociatedControlID="uxCouponCode"
CssClass="span-3 last" runat="server" Text="* Coupon Code:" />
        <ektronUI:TextField  ID="uxCouponCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetCouponId"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            CouponManager couponManager = new CouponManager();
            long couponId = 0;

            couponId = couponManager.GetCouponId(uxCouponCode.Text);

            if (couponId > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Code for the Coupon is " +
couponId , Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Code "+uxCouponCode.Text
+"does not exists", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieve the single coupon item by ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `couponId`. ID of the coupon to retrieved.

Returns

Returns the CouponData object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxCouponIDLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCode" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxExpirationDate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxExpiriationDate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDiscountType" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
```

```
        {
            CouponManager couponManager = new CouponManager();
            long couponId=long.Parse(uxCouponId.Text);
            CouponData couponData = couponManager.GetItem(couponId);

            if (couponData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for coupon with ID "
+ couponData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxCode.Text = "Code : " + couponData.Code;
                uxDescription.Text = "Description : " + couponData.Description;
                uxIsCombineable.Text = "IsCombineable : " + couponData.IsCombinable;
                uxExpirationDate.Text = "ExpirationDate : " + couponData.ExpirationDate;
                uxDiscountType.Text = "Discount Type : "  + couponData.DiscountType;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon "+couponId + "does not
exist. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.CouponCriteria)
```

Retrieves the list of coupon item based on the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Coupon Property
- * Object Value

Parameters

- `criteria`. Criteria used to retrieve the coupon.

Returns

Returns the list of CouponData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCouponPropertyLabel"
AssociatedControlID="uxCouponProperty" CssClass="span-4 last" runat="server"
Text="CouponProperty:" />
        <asp:DropDownList ID="uxCouponProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Code</asp:ListItem>
            <asp:ListItem>Currency Id </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCouponListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Code
                    </th>
                    <th>
                        Description
                    </th>
                    <th>
                        Expiration Date
                    </th>
                    <th>
                        IsActive
                    </th>
                </tr>
            </thead>
```

```
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Code")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Description")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ExpirationDate")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsActive")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            CouponCriteria criteria = new CouponCriteria();

            if (uxCouponProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(CouponProperty.Id,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxCouponProperty.SelectedItem.Text == "Code")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CouponProperty.Code,
```

```
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CouponProperty.CurrencyId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            CouponManager couponManger = new CouponManager();
            List<CouponData> couponList = couponManger.GetList(criteria);

            uxCouponListView.Visible = true;
            uxCouponListView.DataSource = couponList;
            uxCouponListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetProductList

```
GetProductList(Ektron.Cms.Commerce.CouponEntryCriteria)
```

Retrieves the list of product coupon item based on the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object Value

Parameters

- `criteria`. Criteria used to retrieve the product coupon entry.

Returns

Returns the list of product CouponEntryData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCouponEntryPropertyLabel"
AssociatedControlID="uxCouponEntryProperty" CssClass="span-4 last" runat="server"
Text="CouponEntryProperty:" />
        <asp:DropDownList ID="uxCouponEntryProperty" runat="server">
            <asp:ListItem>Coupon Id</asp:ListItem>
            <asp:ListItem>Object ID</asp:ListItem>
            <asp:ListItem>Title </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCouponEntryListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Coupon Id
                    </th>
                    <th>
                        Object Id
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
                        EntryType
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
```

```
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("CouponId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("EntryType")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            CouponEntryCriteria criteria = new CouponEntryCriteria();


            if (uxCouponEntryProperty.SelectedItem.Text == "Coupon Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(CouponEntryProperty.CouponId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxCouponEntryProperty.SelectedItem.Text == "Object Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(CouponEntryProperty.ObjectId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
```

```
                criteria.AddFilter(CouponEntryProperty.Title,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            CouponManager couponManger = new CouponManager();
            List<ProductCouponEntryData> couponEntryList = couponManger.GetProductList
(criteria);

            uxCouponEntryListView.Visible = true;
            uxCouponEntryListView.DataSource = couponEntryList;
            uxCouponEntryListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTaxonomyList

```
GetTaxonomyList(Ektron.Cms.Commerce.CouponEntryCriteria)
```

Retrieves the list of coupon entry item based on the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object Value

Parameters

- `criteria`. Criteria used to retrieve the coupon entry.

Returns

Returns the list of CouponEntryData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCouponEntryPropertyLabel"
AssociatedControlID="uxCouponEntryProperty" CssClass="span-4 last" runat="server"
```

```
Text="CouponEntryProperty:" />
        <asp:DropDownList ID="uxCouponEntryProperty" runat="server">
            <asp:ListItem>Coupon Id</asp:ListItem>
            <asp:ListItem>Object ID</asp:ListItem>
            <asp:ListItem>Title </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCouponEntryListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Coupon Id
                    </th>
                    <th>
                        Object Id
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
                        Path
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("CouponId")%>
            </td>
            <td class="devsite-method">
```

```
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Path")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            CouponEntryCriteria criteria = new CouponEntryCriteria();

            if (uxCouponEntryProperty.SelectedItem.Text == "Coupon Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(CouponEntryProperty.CouponId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxCouponEntryProperty.SelectedItem.Text == "Object Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(CouponEntryProperty.ObjectId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CouponEntryProperty.Title,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            CouponManager couponManger = new CouponManager();
            List<CouponEntryData> couponEntryList = couponManger.GetTaxonomyList
(criteria);
```

```
            uxCouponEntryListView.Visible = true;
            uxCouponEntryListView.DataSource = couponEntryList;
            uxCouponEntryListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IsCouponApplicabletoSubscriptions

```
IsCouponApplicabletoSubscriptions(System.Int64)
```

Checks whether the coupon is applicable to the subscription.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID

Parameters

- `couponId`. ID of the coupon.

Returns

Returns true, if the coupon is applicable to the subscriptions; otherwise false.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="IsCouponApplicable"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
```

```
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            long couponId = long.Parse(uxCouponId.Text);

            bool result = couponManager.IsCouponApplicabletoSubscriptions(couponId);
            if (result)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
applicable to the subscription", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
not applicable to the subscription", Message.DisplayModes.Information);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

IsCouponAppliedToObject

```
IsCouponAppliedToObject
(System.Int64,System.Int64,Ektron.Cms.Common.EkEnumeration.CMSObjectTypes)
```

Checks whether the coupon is applied to the particular object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID
- * Object ID
- Object Type

Parameters

- `couponId`. ID of the coupon.
- `objectId`. ID of the object.
- `objectType`. CmsObjectTypes enum.

Returns

Returns true, if the coupon is applied to the object; otherwise false.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
      <li class="clearfix">
        <ektronUI:Label ID="uxObjectIdLabel" AssociatedControlID="uxObjectId"
CssClass="span-3 last" runat="server" Text="* Object Id:" />
        <ektronUI:TextField  ID="uxObjectId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponTypeLabel" AssociatedControlID="uxCouponType"
CssClass="span-3 last" runat="server" Text=" Coupon Type:" />
         <asp:DropDownList ID="uxCouponType" runat="server">
            <asp:ListItem Value="0">Content</asp:ListItem>
            <asp:ListItem Value="1">Folder</asp:ListItem>
            <asp:ListItem Value="2">User</asp:ListItem>
            <asp:ListItem Value="3">UserGroup</asp:ListItem>
            <asp:ListItem Value ="4">Library</asp:ListItem>
            <asp:ListItem Value="5">Collection</asp:ListItem>
            <asp:ListItem Value="6">Menu</asp:ListItem>
            <asp:ListItem Value="7">Calendar</asp:ListItem>
            <asp:ListItem Value="8">Subscription</asp:ListItem>
            <asp:ListItem Value="9">Notification</asp:ListItem>
```

```
            <asp:ListItem Value="10">Blog</asp:ListItem>
            <asp:ListItem Value="11">BlogPost</asp:ListItem>
            <asp:ListItem Value="12">BlogComment</asp:ListItem>
            <asp:ListItem Value="13">DiscussionBoard</asp:ListItem>
            <asp:ListItem Value="14">DiscussionForum</asp:ListItem>
            <asp:ListItem Value="15">RestrictIP</asp:ListItem>
            <asp:ListItem Value="16">ReplaceWord</asp:ListItem>
            <asp:ListItem Value="17">UserRank</asp:ListItem>
            <asp:ListItem Value="18">DiscussionTopic</asp:ListItem>
            <asp:ListItem Value="19">CommunityGroup</asp:ListItem>
            <asp:ListItem Value="20">TaxonomyNode</asp:ListItem>
            <asp:ListItem Value="21">CatalogEntry</asp:ListItem>
            <asp:ListItem Value="22">MessageBoard</asp:ListItem>
            <asp:ListItem Value="23">CalendarEvent</asp:ListItem>
            <asp:ListItem Value="24">MicroMessage</asp:ListItem>
            <asp:ListItem Value="25">Subscriber</asp:ListItem>
        </asp:DropDownList>
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="IsCouponApplied"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            long couponId = long.Parse(uxCouponId.Text);
            long objectId = long.Parse(uxObjectId.Text);
            Ektron.Cms.Common.EkEnumeration.CMSObjectTypes objectType=
(Ektron.Cms.Common.EkEnumeration.CMSObjectTypes) int.Parse(uxObjectType.SelectedValue);

            bool result = couponManager.IsCouponAppliedToObject(couponId, objectId,
objectType);
            if (result)
```

```
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
applied to the object", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
not applied to the object", Message.DisplayModes.Information);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IsCouponUsedForBasket

```
IsCouponUsedForBasket(System.Int64)
```

Checks whether the coupon is used in the basket.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID

Parameters

- `couponId`. ID of the coupon.

Returns

Returns true, if the coupon is used in any basket; otherwise false.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="IsCouponUsed"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            long couponId = long.Parse(uxCouponId.Text);

            bool result = couponManager.IsCouponUsedForBasket(couponId);
            if (result)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
used in the basket", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
not used in the basket", Message.DisplayModes.Information);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

IsCouponUsedForOrder

```
IsCouponUsedForOrder(System.Int64)
```

Checks whether the coupon is used in any order.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID

Parameters

- `couponId`. ID of the coupon.

Returns

Returns true, if the coupon is used in order; otherwise false.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="IsCouponUsed"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            long couponId = long.Parse(uxCouponId.Text);

            bool result = couponManager.IsCouponUsedForOrder(couponId);
            if (result)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
used in the order", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
not used in the order", Message.DisplayModes.Information);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

SaveCouponApplications

```
SaveCouponApplications(System.Int64,System.Collections.Generic.List)
```

Saves all the applications (taxonomy, catalog, product) for a given coupon. This is assumed to be an all inclusive lists. Any existing coupon applications that are not included are deleted.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID
- * Object ID
- Object Type

Parameters

- `couponId`. ID of coupon to update.
- `couponEntryList`. List of all CouponEntryData applications for the given coupon.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
      <li class="clearfix">
        <ektronUI:Label ID="uxObjectIdLabel" AssociatedControlID="uxObjectId"
CssClass="span-3 last" runat="server" Text="* Object Id:" />
        <ektronUI:TextField  ID="uxObjectId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponTypeLabel" AssociatedControlID="uxCouponType"
CssClass="span-3 last" runat="server" Text=" Coupon Type:" />
         <asp:DropDownList ID="uxCouponType" runat="server">
            <asp:ListItem Value="1">Folder</asp:ListItem>
            <asp:ListItem Value="20">TaxonomyNode</asp:ListItem>
            <asp:ListItem Value="21">CatalogEntry</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIncludedLabel" AssociatedControlID="uxCouponActive"
CssClass="span-3 last" runat="server" Text=" Is Active:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxCouponActive"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="SaveCoupon Application"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
```

```
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            bool isInclude = false;
            long couponId = long.Parse(uxCouponId.Text);
            List<CouponEntryData> listCouponEntry = new List<CouponEntryData>();
            if (uxCouponInclude.Checked)
            {
                isInclude = true;
            }
            CouponEntryData couonEntryData = new CouponEntryData()
            {
                CouponId = couponId ,
                DataState = Ektron.Cms.Common.EkEnumeration.DataState.Added,
                ObjectType = (Ektron.Cms.Common.EkEnumeration.CMSObjectTypes)int.Parse
(uxObjectType.SelectedValue),
                ObjectId = long.Parse(uxObjectId.Text),
                IsIncluded = isInclude
            };
            listCouponEntry.Add(couonEntryData);
            couponManager.SaveCouponApplications(couponId, listCouponEntry);
            MessageUtilities.UpdateMessage(uxMessage, "Coupon application is saved" +
couonEntryData.ObjectId, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Update

```
Update(Ektron.Cms.Commerce.CouponData)
```

Updates the existing coupon data object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID
- Coupon Description
- Discount Value

Parameters

- `couponData`. CouponData object to be updated.

Returns

Updated coupon data is returned.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponDescriptionLabel"
AssociatedControlID="uxCouponDescription" CssClass="span-3 last" runat="server" Text="
Coupon Description:" />
        <ektronUI:TextField  ID="uxCouponDescription" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDiscountLable" AssociatedControlID="uxDiscountValue"
CssClass="span-3 last" runat="server" Text=" Discount Value:" />
        <ektronUI:TextField  ID="uxDiscountValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            long counponId=long.Parse(uxCouponId.Text);
            CouponData couponData = couponManager.GetItem(counponId);
            if (couponData != null)
            {
                couponData.Description = uxCouponDescription.Text;
                couponData.DiscountValue = Decimal.Parse(uxDiscountValue.Text);
                couponManager.Update(couponData);

                MessageUtilities.UpdateMessage(uxMessage, "Coupon updated with Id " +
couponData.Id, Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);


        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Validate (coupon)

```
Validate(System.Int64)
```

Validate the coupon by coupon ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID

Parameters

- `couponId`. ID of the coupon to validate.

Returns

Returns true if validation is true; otherwise false.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Validate"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            long couponId = long.Parse(uxCouponId.Text);

            bool result = couponManager.Validate(couponId);
            if (result)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
valid", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
not valid", Message.DisplayModes.Information);
            }
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Validate (product)

```
Validate(System.Int64,System.Int64)
```

Validate the coupon by coupon id and product ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Coupon ID
- * Product ID

Parameters

- `couponId`. ID of the coupon.
- `productId`. ID of the product.

Returns

Returns true if validation is true; otherwise false.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxProductIdLabel" AssociatedControlID="uxProductId"
CssClass="span-3 last" runat="server" Text="* Product Id:" />
        <ektronUI:TextField  ID="uxProductId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Validate"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CouponManager couponManager = new CouponManager();
            long couponId = long.Parse(uxCouponId.Text);
            long productId = long.Parse(uxProductId.Text);

            bool result = couponManager.Validate(couponId,productId);
            if (result)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
valid", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Coupon " + couponId + " is
not valid", Message.DisplayModes.Information);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);


        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Data Classes

CouponCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `CouponCriteria()`

  ```
  public CouponCriteria()
  ```

- `CouponCriteria(Ektron.Cms.Commerce.CouponProperty, EkEnumeration.OrderByDirection)`

  ```
  public
    CouponCriteria(Ektron.Cms.Commerce.CouponProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `CouponProperty` are:

  - AppliesToQuantity
  - ApplyOneTime
  - Code
  - CouponType
  - CurrencyId
  - Description
  - DiscountType
  - DiscountValue
  - ExpirationDate
  - Id
  - IsActive
  - IsApplicabletoSubscriptions
  - IsCombinable
  - IsRedeemable
  - MaximumAmount
  - MaximumUses
  - MinimumAmount
  - StartDate
  - UseCount

CouponData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `AppliesToQuantity`. For an DiscountType.Amount coupon, defines the applicable quantity for a single line in a basket. Default is 1. Example, if you have a $5 Amount coupon, and a quantity 5 of a $10 item (total cost $50), by default the coupon applies to the first item only: $50 - $5 = $45. Change the AppliesToQuantity value to 2, the coupon applies to 2 of the item quantity: $50 - $10 = $40. Change the AppliesToQuantity value to 3, the coupon applies to 3 of the item quantity: $50 - $15 = $35. And so on...

  ```
  public bool AppliesToQuantity { set; get; }
  ```

- `Code`. Gets or sets the coupon code.

  ```
  public string Code { set; get; }
  ```

- `CouponData()`

  ```
  public CouponData()
  ```

- `CouponData(long, string, System.DateTime, System.DateTime, EkEnumeration.CouponDiscountType, decimal, decimal, EkEnumeration.CouponType, int, bool, int).`

  ```
  public CouponData(long couponId, string couponCode,
    System.DateTime startDate, System.DateTime expirationDate,
    EkEnumeration.CouponDiscountType discountType,
    decimal discountValue, decimal minimumAmount,
    EkEnumeration.CouponType couponType, int maximumUses,
    bool applyOnePerCustomer, int currencyId)
  ```

- `CouponType`. Gets or sets the coupon type.

  ```
  public EkEnumeration.CouponType CouponType { set; get; }
  ```

- `CreateFixedDiscountCoupon(string, System.DateTime, System.DateTime, decimal, decimal, EkEnumeration.CouponType, int, bool, int)`

  ```
  public static Ektron.Cms.Commerce.CouponData
    CreateFixedDiscountCoupon(string couponCode,
      System.DateTime startDate, System.DateTime
      expirationDate, decimal discountAmount,
      decimal minimumAmount, EkEnumeration.CouponType
      couponType, int maximumUses, bool applyOnePerCustomer,
      int currencyId)
  ```

- `CreateFreeShippingCoupon(string, System.DateTime, System.DateTime, decimal, EkEnumeration.CouponType, int, bool)`

  ```
  public static Ektron.Cms.Commerce.CouponData
    CreateFreeShippingCoupon(string couponCode,
      System.DateTime startDate, System.DateTime expirationDate,
      decimal minimumAmount, EkEnumeration.CouponType couponType,
      int maximumUses, bool applyOnePerCustomer)
  ```

- `CreatePercentageDiscountCoupon(string, System.DateTime, System.DateTime, decimal, decimal, EkEnumeration.CouponType, int, bool, int)`

```
public static Ektron.Cms.Commerce.CouponData
  CreatePercentageDiscountCoupon(string couponCode,
    System.DateTime startDate, System.DateTime expirationDate,
    decimal discountPercentage, decimal minimumAmount,
    EkEnumeration.CouponType couponType, int maximumUses,
    bool applyOnePerCustomer, int currencyId)
```

- `CurrencyId`. Gets or sets the Id of the currency this coupon applies to.

```
public int CurrencyId { set; get; }
```

- `Description`. Gets or sets the description of the coupon.

```
public string Description { set; get; }
```

- `DiscountType`. Gets or sets the Coupon Discount Type.

```
public EkEnumeration.CouponDiscountType DiscountType { set; get; }
```

- `DiscountValue`. Gets or sets the value of the discount. For a fixed amount, this would be the amount (i.e. 5.00). For percentage, it would be the discount percent value in whole numbers (i.e. 15% = 15.00).

```
public decimal DiscountValue { set; get; }
```

- `ExpirationDate`. Gets or sets the date the coupon expires.

```
public System.DateTime ExpirationDate { set; get; }
```

- `Id`. Gets or sets the Id of the coupon.

```
public long Id { set; get; }
```

- `IsActive`. Determines whether or not this coupon is currently active. Deleted coupons remain in the system and are marked inactive.

```
public bool IsActive { set; get; }
```

- `IsApplicabletoSubscription`. Determines whether this coupon is applicable to subscription.

```
public bool IsApplicabletoSubscriptions { set; get; }
```

- `IsCombinable`. Determines whether or not this coupon can be combined with other coupons. If true, this coupon can be combined with other "Combinable" coupons.

```
public bool IsCombinable { set; get; }
```

- `IsRedeemable`. Determines whether or not this coupon is currently redeemable. A coupon may not be reddemable if the it is inactive, not within its valid period, or has been redeemed the maximum number of times.

```
public bool IsRedeemable { set; get; }
```

- `MaximumAmount`. Gets or sets the maximum order amount that this coupon can be applied to.

```
 public decimal MaximumAmount { set; get; }
```

- `MaximumUses`. Gets or sets the maximim number of times this coupon can be used before it expires.

```
public int MaximumUses { set; get; }
```

- `MinimumAmount`. Gets or sets the minimum order amount required before the coupon is applicable.

```
public decimal MinimumAmount { set; get; }
```

- `OnePerCustomer`. Gets or sets the OnePerCuster value. If true, this coupon can only be used once by each customer.

```
public bool OnePerCustomer { set; get; }
```

- `StartDate`. Gets or sets the date the coupon becomes valid.

```
public System.DateTime StartDate { set; get; }
```

- `UserCount`. Returns the count representing the number of orders that this coupon has been applied to.

```
public long UseCount { set; get; }
```

- `Validate`. Validates the CouponData object is valid for saving. Returns a ValidationResult object with any validation errors.

```
public ValidationResults Validate()
```

CouponEntryCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `CouponEntryCriteria()`

```
public CouponEntryCriteria()
```

- `CouponEntryCriteria(Ektron.Cms.Commerce.CouponEntryProperty, EkEnumeration.OrderByDirection)`

```
public
  CouponEntryCriteria(Ektron.Cms.Commerce.CouponEntryProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

  Accepted fields for the `CouponEntryProperty` are:

  - CouponId
  - IsIncluded
  - ObjectId
  - ObjectType
  - Path
  - Title

CouponEntryData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `Clone()`

```
public virtual Ektron.Cms.Commerce.CouponEntryData Clone()
```

- `CouponEntryData()`

```
public CouponEntryData()
```

- `CouponEntryData(long, EkEnumeration.CMSObjectTypes, long, string, bool)`. Gets or sets the Id of the coupon.

```
public CouponEntryData(long objectId,
  EkEnumeration.CMSObjectTypes objectType,
  long couponId, string title, bool isIncluded)
```

- `CouponId`. Gets or sets the ID of the coupon.

```
public long CouponId { set; get; }
```

- `DataState`. Gets or sets the current saved state for the CouponEntry. Used by the Cms internally.

```
public EkEnumeration.DataState DataState { set; get; }
```

- `IsIncluded`. Gets or sets whether or not the Object is included in the coupon. If true, the object is included in the coupon. If false, the coupon is NOT included in the coupon.

```
public bool IsIncluded { set; get; }
```

- `ObjectId`. Gets or sets the Id of object the coupon is applied to.

```
public long ObjectId { set; get; }
```

- `ObjectType`. Gets or sets the type of object this coupon is applied to.

```
public EkEnumeration.CMSObjectTypes ObjectType { set; get; }
```

- `Path`. Gets or sets the Path of the object.

```
public string Path { set; get; }
```

- `Title`. Gets or sets the Title of the object.

```
public string Title { set; get; }
```

CreditCardTypeManager

9.00 and higher

The class manages credit card types in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.CreditCardTypeManager
```

Constructors

- `CreditCardTypeManager()`
- `CreditCardTypeManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Commerce.CreditCardTypeData)
```

Adds a new credit card type to CMS. The ID property is populated with the new credit card type's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Credit Card Type Name
- * Regex (/^3[4,7][0-9]{13}$/)
- Is Accepted

Parameters

- `item`. Credit card type object to add.

Returns

Returns added CreditCardTypeData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-6
last" runat="server" Text="* Credit Card Type Name:" />
        <ektronUI:TextField ID="uxName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"Text="0" />
    </li>
     <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegexLabel" AssociatedControlID="uxRegex" CssClass="span-6
last" runat="server" Text="* Regex(/^3[4,7][0-9]{13}$/):" />
        <ektronUI:TextField ID="uxRegex" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsAcceptedLabel" AssociatedControlID="uxIsAccepted"
CssClass="span-6 last" runat="server" Text=" IsAccepted:" />
        <asp:checkbox ID="uxIsAccepted" runat="server" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            CreditCardTypeManager ccTypeManager = new CreditCardTypeManager();
            CreditCardTypeData ccTypeData = new CreditCardTypeData();
            ccTypeData.Name = uxName.Text;
            ccTypeData.Regex = uxRegex.Text;
            ccTypeData.IsAccepted = uxIsAccepted.Checked;
            ccTypeManager.Add(ccTypeData);
            MessageUtilities.UpdateMessage(uxMessage, "CreditCardTypeData is added with
Id " + ccTypeData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a credit card type from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of credit card type to delete.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCreditCardTypeIdLabel"
AssociatedControlID="uxCreditCardTypeId" CssClass="span-4 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField ID="uxCreditCardTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxCreditCardTypeId.Text);

            CreditCardTypeManager ccTypeManager = new CreditCardTypeManager();
            ccTypeManager.Delete(ID);
            MessageUtilities.UpdateMessage(uxMessage, "CreditCardType with Id " +
uxCreditCardTypeId.Text + " deleted", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetAcceptedCreditCardList

```
GetAcceptedCreditCardList()
```

Retrieves a list of accepted credit card types.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Returns

Return the accepted credit card type data list.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetAcceptedCreditCardList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCreditCardTypeDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >No records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                        <th>
                        Name
                    </th>
                    <th>
                        Regex
                    </th>
                    <th>
                        IsAccepted
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
```

```
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Regex")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsAccepted")%>
            </td>

        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CreditCardTypeManager ccTypeManager = new CreditCardTypeManager();
            List<CreditCardTypeData> CCTypeDataList = new List<CreditCardTypeData>();

            CCTypeDataList = ccTypeManager.GetAcceptedCreditCardList();
            uxCreditCardTypeDataListView.Visible = true;
            uxCreditCardTypeDataListView.DataSource = CCTypeDataList;
            uxCreditCardTypeDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a credit card type by ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of credit card type to retrieve.

Returns

Return the CreditCardTypeData.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCreditCardTypeIdLabel"
AssociatedControlID="uxCreditCardTypeId" CssClass="span-3 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField ID="uxCreditCardTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
    </ol>
    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxRegex" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsAccepted" runat="server"></asp:Literal>
```

```
    </li>
  </ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long CreditCardTypeId = long.Parse(uxCreditCardTypeId.Text);

            CreditCardTypeManager ccTypeManager = new CreditCardTypeManager();
            CreditCardTypeData ccTypeData = new CreditCardTypeData();
            ccTypeData=ccTypeManager.GetItem(CreditCardTypeId);
            if (ccTypeData != null)
            {

                MessageUtilities.UpdateMessage(uxMessage, "Details for Credit Card Type
Id " + ccTypeData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxName.Text = "Name : " + ccTypeData.Name;
                uxRegex.Text = "Regex : " + ccTypeData.Regex;
                uxIsAccepted.Text = "IsAccepted : " + ccTypeData.IsAccepted;

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either Credit Card Type with
ID " + CreditCardTypeId + " does not exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.CreditCardTypeCriteria)
```

Retrieves a list of credit card types.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Credit Card Type Property
- * Object Value

Parameters

- `criteria`. Criteria object used for sorting, paging, and filtering results.

Returns

Return the CreditCardTypeData list.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCreditCardTypePropertyLabel"
AssociatedControlID="uxCreditCardType" CssClass="span-5 last" runat="server" Text="*
CreditCardType Property:" />
        <asp:DropDownList ID="uxCreditCardType" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxCreditCardTypeDataListView" runat="server"
```

```
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >No records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                        <th>
                        Name
                    </th>
                    <th>
                        Regex
                    </th>
                    <th>
                        IsAccepted
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Regex")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsAccepted")%>
            </td>
      </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            CreditCardTypeManager ccTypeManager = new CreditCardTypeManager();
            CreditCardTypeCriteria ccTypeCri = new CreditCardTypeCriteria();

            if (uxCreditCardType.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                ccTypeCri.AddFilter(CreditCardTypeProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                ccTypeCri.AddFilter(CreditCardTypeProperty.Name,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            List<CreditCardTypeData> ccTypeDataList = ccTypeManager.GetList(ccTypeCri);

            uxCreditCardTypeDataListView.Visible = true;
            uxCreditCardTypeDataListView.DataSource = ccTypeDataList;
            uxCreditCardTypeDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IsCardValid

```
IsCardValid(Ektron.Cms.Commerce.CreditCardPayment,System.Boolean)
```

Validates credit card values and date. Does not communicate with the payment gateway.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * CCID
- * Card Number
- * Card Type
- * Expiration Date
- Required CCID

Parameters

- `creditcard`. Credit card payment details.
- `requireCCID`. True or false for required CCID (credit card ID) value.

Returns

Returns the boolean value.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
    <ektronUI:Label ID="uxCCIDLabel" AssociatedControlID="uxCCID" CssClass="span-4 last"
runat="server" Text="* CCID:" />
    <ektronUI:TextField ID="uxCCID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Label ID="uxNumberLabel" AssociatedControlID="uxNumber" CssClass="span-4
last" runat="server" Text="* Card Number :" />
    <ektronUI:TextField ID="uxNumber" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxTypeNameLabel" AssociatedControlID="uxTypeName"
CssClass="span-4 last" runat="server" Text="* Card Type Name:" />
    <ektronUI:TextField ID="uxTypeName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Label ID="uxDateLabel" AssociatedControlID="uxDate" CssClass="span-4 last"
runat="server" Text="*ExpirationDate :" />
    <ektronUI:DatePicker Rows="3" ID="uxDate" CssClass="span-6" runat="server"
Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Label ID="uxRequiredCCIDLabel" AssociatedControlID="uxRequiredCCID"
CssClass="span-4 last" runat="server" Text=" RequiredCCID:" />
    <asp:checkbox ID="uxRequiredCCID" runat="server" />
</li>
```

```
<li class="clearfix">
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="IsCardValid"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            bool value;
            DateTime ExpirationDate = new DateTime();
            ExpirationDate = Convert.ToDateTime(uxDate.Text.ToString());
            CCExpirationDate expDate = new CCExpirationDate
            {
                Date = ExpirationDate.Date,
                Month =
(Ektron.Cms.Common.EkEnumeration.CCExpirationMonth)ExpirationDate.Month,
                Year = ExpirationDate.Year,

            };

            CreditCardTypeManager ccTypeManager = new CreditCardTypeManager();
            CreditCardPayment ccPayment = new CreditCardPayment();
            ccPayment.CCID = uxCCID.Text;
            ccPayment.Number  = uxNumber.Text;
            ccPayment.TypeName = uxTypeName.Text;
            ccPayment.ExpirationDate = expDate;
            value = ccTypeManager.IsCardValid(ccPayment,uxRequiredCCID.Checked);
            if(value)
                MessageUtilities.UpdateMessage(uxMessage, "The given card is valid ",
Message.DisplayModes.Success);
            else
                MessageUtilities.UpdateMessage(uxMessage, "The given card is not valid
", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IsDateValid

```
IsDateValid(Ektron.Cms.Commerce.CreditCardPayment)
```

Validates the date of a credit card versus the server's date-and-time.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * CCID
- * Card Number
- * Card Type Name
- * Expiration Date

Parameters

- `creditcard`. Credit card payment details.

Returns

Returns the boolean value.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
    <ektronUI:Label ID="uxCCIDLabel" AssociatedControlID="uxCCID" CssClass="span-4 last"
runat="server" Text="* CCID:" />
    <ektronUI:TextField ID="uxCCID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Label ID="uxNumberLabel" AssociatedControlID="uxNumber" CssClass="span-4
last" runat="server" Text="* Card Number :" />
    <ektronUI:TextField ID="uxNumber" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxTypeNameLabel" AssociatedControlID="uxTypeName"
CssClass="span-4 last" runat="server" Text="* Card Type Name:" />
    <ektronUI:TextField ID="uxTypeName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
```

```
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Label ID="uxDateLabel" AssociatedControlID="uxDate" CssClass="span-4 last"
runat="server" Text="*ExpirationDate :" />
    <ektronUI:DatePicker Rows="3" ID="uxDate" CssClass="span-6" runat="server"
Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
</li>
<li class="clearfix">
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="IsDateValid"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            bool value;
            DateTime ExpirationDate = new DateTime();
            ExpirationDate = Convert.ToDateTime(uxDate.Text.ToString());
            CCExpirationDate expDate = new CCExpirationDate
            {
                Date = ExpirationDate.Date,
                Month =
(Ektron.Cms.Common.EkEnumeration.CCExpirationMonth)ExpirationDate.Month,
                Year = ExpirationDate.Year,

            };
            CreditCardTypeManager ccTypeManager = new CreditCardTypeManager();
            CreditCardPayment ccPayment = new CreditCardPayment();
            ccPayment.CCID = uxCCID.Text;
            ccPayment.Number = uxNumber.Text;
            ccPayment.TypeName = uxTypeName.Text;
            ccPayment.ExpirationDate = expDate;
            value = ccTypeManager.IsDateValid(ccPayment);
            if (value)
                MessageUtilities.UpdateMessage(uxMessage, "The given card date is valid
", Message.DisplayModes.Success);
            else
                MessageUtilities.UpdateMessage(uxMessage, "The given card date is not
valid ", Message.DisplayModes.Error);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.CreditCardTypeData)
```

Updates an existing credit card type in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Credit Card Type Name
- * Regex (/^3[4,7][0-9]{13}$/)
- Is Accepted

Parameters

- `item`. Credit card type object to save.

Returns

Returns an updated CreditCardTypeData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCreditCardTypeIdLabel"
AssociatedControlID="uxCreditCardTypeId" CssClass="span-6 last" runat="server" Text="
Id:" />
        <ektronUI:TextField ID="uxCreditCardTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-6
last" runat="server" Text="* Credit Card Type Name:" />
        <ektronUI:TextField ID="uxName" CssClass="span-6" runat="server"
```

```
ValidationGroup="RegisterValidationGroup" Text="" />
    </li>
    <div class="ektronTopSpace"></div>
     <li class="clearfix">
        <ektronUI:Label ID="uxRegexLabel" AssociatedControlID="uxRegex" CssClass="span-6
last" runat="server" Text="* Regex (/^3[4,7][0-9]{13}$/):" />
        <ektronUI:TextField ID="uxRegex" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
    </li>
    <div class="ektronTopSpace"></div>
     <li class="clearfix">
        <ektronUI:Label ID="uxIsAcceptedLabel" AssociatedControlID="uxIsAccepted"
CssClass="span-6 last" runat="server" Text=" IsAccepted:" />
        <asp:checkbox ID="uxIsAccepted" runat="server" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long CreditCardTypeId = long.Parse(uxCreditCardTypeId.Text);
            CreditCardTypeManager ccTypeManager = new CreditCardTypeManager();
            CreditCardTypeData ccTypeData = new CreditCardTypeData();
            ccTypeData = ccTypeManager.GetItem(CreditCardTypeId);
            if (CreditCardTypeId > 0 && ccTypeData!=null)
            {
                ccTypeData.Name = uxName.Text;
                ccTypeData.Regex = uxRegex.Text;
                ccTypeData.IsAccepted = uxIsAccepted.Checked;
                ccTypeManager.Update(ccTypeData);
                MessageUtilities.UpdateMessage(uxMessage, "CreditCardType Updated with
Id " + uxCreditCardTypeId.Text, Message.DisplayModes.Success);
```

```
            }
            else
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid
CreditCardType ID.", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

CreditCardTypeCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `CreditCardTypeCriteria()`

  ```
  public CreditCardTypeCriteria()
  ```

- `CreditCardTypeCriteria (Ektron.Cms.Commerce.CreditCardTypeProperty, EkEnumeration.OrderByDirection)`

  ```
  public CreditCardTypeCriteria(Ektron.Cms.Commerce.CreditCardTypeProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `CreditCardTypeProperty` are:

  - Id
  - IsAccepted
  - Name

CreditCardTypeData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `Id`. Gets or sets the Id of the credit card type.

```
public long Id { set; get; }
```

- `Image`. Gets or sets path to image for credit card type.

```
public string Image { set; get; }
```

- `IsAccepted`. Gets or sets whether or not the credit car type is accepted.

```
public bool IsAccepted { set; get; }
```

- `Name`. Gets or sets the name of the credit card.

```
public string Name { set; get; }
```

- `Regex`. Gets or sets the Regex validation for the credit card type.

```
public string Regex { set; get; }
```

CurrencyManager

9.00 and higher

The class manages commerce currencies in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.CurrencyManager
```

Constructors

- `CurrencyManager()`
- `CurrencyManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *CanDelete* on page 502
- *Delete* on page 504
- *GetActiveCurrencyList* on page 505
- *GetCurrentCurrency* on page 507
- *GetDefaultCurrency* on page 509
- *GetItem* on page 510
- *GetList* on page 512
- *Update* on page 515

Add

```
Add(Ektron.Cms.Commerce.CurrencyData)
```

Adds a new currency to the CMS. The Currency.Id property is populated with the new Currency's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Alpha ISO Code
- Name
- Culture Code
- Enabled

Parameters

- `currencyData`. CurrencyData object to be added.

Returns

Returns the CurrencyData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyIDLabel" AssociatedControlID="uxCurrencyId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField  ID="uxCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyNameLabel" AssociatedControlID="uxCurrencyName"
CssClass="span-3 last" runat="server" Text=" Name:" />
        <ektronUI:TextField  ID="uxCurrencyName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyAlbhaLabel"
AssociatedControlID="uxCurrencyAlphaCode" CssClass="span-3 last" runat="server" Text="*
AlphaIsoCode:" />
        <ektronUI:TextField  ID="uxCurrencyAlphaCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyCultureCodeLabel"
AssociatedControlID="uxCurrencyCultureCode" CssClass="span-3 last" runat="server" Text="
Culture Code:" />
```

```
        <ektronUI:TextField  ID="uxCurrencyCultureCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyEnabledLabel"
AssociatedControlID="uxCurrencyEnabled" CssClass="span-3 last" runat="server" Text="
Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxCurrencyEnabled"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CurrencyManager currencyManager = new CurrencyManager();
            bool isEnabled = false;
            if (uxCurrencyEnabled.Checked)
            {
                isEnabled = true;
            }
            CurrencyData currencyData = new CurrencyData()
            {
                AlphaIsoCode = uxCurrencyAlphaCode.Text,
                CultureCode = uxCurrencyCultureCode.Text,
                Id = int.Parse(uxCurrencyId.Text),
                Name = uxCurrencyName.Text,
                Enabled = isEnabled
            };

            currencyData = currencyManager.Add(currencyData);
            if (currencyData.Id > 0)
```

```
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Currency Added with Id " +
currencyData.Id , Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Currency adding process is
failed", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

CanDelete

```
CanDelete(System.Int32)
```

Checks to see if a currency can be deleted.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Currency ID

Parameters

- `currencyData`. Currency data object to be added.

Returns

CurrencyData object.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyIdLabel" AssociatedControlID="uxCurrencyId"
CssClass="span-3 last" runat="server" Text="* Currency Id:" />
        <ektronUI:TextField  ID="uxCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="CanDelete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using Microsoft.Practices.EnterpriseLibrary.Validation;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CurrencyManager currencyManager = new CurrencyManager();
            int CurrencyId = int.Parse(uxCurrencyId.Text);


            bool delete = currencyManager.CanDelete(CurrencyId);
            if (delete)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Currency " + CurrencyId + "
can be deleted", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Currency " + CurrencyId + "
cannot be deleted",  Message.DisplayModes.Information);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Delete

```
Delete(System.Int32)
```

Deletes a currency from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Currency ID

Parameters

- `currencyId`. ID of currency to be deleted.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyIdLabel" AssociatedControlID="uxCurrencyId"
CssClass="span-3 last" runat="server" Text="* Currency Id:" />
        <ektronUI:TextField  ID="uxCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CurrencyManager currencyManager = new CurrencyManager();
            int currencyId = int.Parse(uxCurrencyId.Text);

            currencyManager.Delete(currencyId);
            MessageUtilities.UpdateMessage(uxMessage, "Currency " + currencyId + " is
deleted", Message.DisplayModes.Success );

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetActiveCurrencyList

```
GetActiveCurrencyList()
```

Gets active currency List

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Returns

List of active currencies.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Active Currency"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCouponEntryListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
```

```
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Id
                    </th>
                    <th>
                       Name
                    </th>
                    <th>
                        AlphaIsoCode
                    </th>
                    <th>
                        Culture Code
                    </th>
                    <th>
                        Enabled
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("AlphaIsoCode")%>
            </td>
            <td class="devsite-method">
                <%# Eval("CultureCode")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Enabled")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CurrencyManager currencyManger = new CurrencyManager();
            List<CurrencyData> currencyDataList = currencyManger.GetActiveCurrencyList
();

            uxCurrencyListView.Visible = true;
            uxCurrencyListView.DataSource = currencyDataList;
            uxCurrencyListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetCurrentCurrency

```
GetCurrentCurrency()
```

Gets current currency.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Returns

Current CurrencyData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Current Currency"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
```

```
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAlpha" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCultureCode" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxEnabled" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CurrencyManager currencyManager = new CurrencyManager();
            CurrencyData currencyData = currencyManager.GetCurrentCurrency();

            if (currencyData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for Current  Currency
with ID " + currencyData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxAlpha.Text = "AlphaIsoCode : " + currencyData.AlphaIsoCode;
                uxCultureCode.Text = "Culture Code : " + currencyData.CultureCode;
                uxName.Text = "Name : " + currencyData.Name;
                uxEnabled.Text = "Enabled : " + currencyData.Enabled;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
```

```
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetDefaultCurrency

```
GetDefaultCurrency()
```

Gets default currency.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Returns

Default CurrencyData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Default Currency"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAlpha" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCultureCode" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxEnabled" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CurrencyManager currencyManager = new CurrencyManager();
            CurrencyData currencyData = currencyManager.GetDefaultCurrency();

            if (currencyData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for Default  Currency
with ID " + currencyData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxAlpha.Text = "AlphaIsoCode : " + currencyData.AlphaIsoCode;
                uxCultureCode.Text = "Culture Code : " + currencyData.CultureCode;
                uxName.Text = "Name : " + currencyData.Name;
                uxEnabled.Text = "Enabled : " + currencyData.Enabled;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetItem

```
GetItem(System.Int32)
```

Retrieves a currency by currency ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Currency ID

Parameters

`currencyId`. ID of currency to be retrieved.

Returns

CurrencyData object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyIDLabel" AssociatedControlID="uxCurrencyId"
CssClass="span-3 last" runat="server" Text="* Currency Id :" />
        <ektronUI:TextField ID="uxCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAlpha" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCultureCode" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxEnabled" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CurrencyManager currencyManager = new CurrencyManager();
            int CurrencyId=int.Parse(uxCurrencyId.Text);
            CurrencyData currencyData = currencyManager.GetItem(CurrencyId);

            if (currencyData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for Currency with ID
" + currencyData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxAlpha.Text = "AlphaIsoCode : " + currencyData.AlphaIsoCode;
                uxCultureCode.Text = "Culture Code : " + currencyData.CultureCode;
                uxName.Text = "Name : " + currencyData.Name;
                uxEnabled.Text = "Enabled : " + currencyData.Enabled;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Currency "+CurrencyId + "
does not exist. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetList

```
GetList(Ektron.Cms.Commerce.CurrencyCriteria)
```

Retrieves a list of currencies.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object Value

Parameters

- `criteria`. Criteria by which to filter currency being retrieved.

Returns

List of currencies meeting criteria filters.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyPropertyLabel"
AssociatedControlID="uxCurrencyProperty" CssClass="span-4 last" runat="server"
Text="Currency Property:" />
        <asp:DropDownList ID="uxCurrencyProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
            <asp:ListItem>AlphaIsoCode </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCouponListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Name
                    </th>
                    <th>
                        AlphaIsoCode
                    </th>
                    <th>
                        Culture Code
                    </th>
                    <th>
```

```
                        Enabled
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("AlphaIsoCode")%>
            </td>
            <td class="devsite-method">
                <%# Eval("CultureCode")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Enabled")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            CurrencyCriteria criteria = new CurrencyCriteria();

            if (uxCurrencyProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = Convert.ToInt32 (uxObjectValue.Text);
                criteria.AddFilter(CurrencyProperty.Id,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
```

```
            else if (uxCurrencyProperty.SelectedItem.Text == "Name")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CurrencyProperty.Name,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CurrencyProperty.AlphaIsoCode ,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            CurrencyManager currencyManger = new CurrencyManager();
            List<CurrencyData> currencyList = currencyManger.GetList(criteria);

            uxCurrencyListView.Visible = true;
            uxCurrencyListView.DataSource = currencyList;
            uxCurrencyListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.CurrencyData)
```

Updates an existing currency in Cms.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- Name
- * Alpha ISO Code
- Culture Code

Parameters

- `currencyData`. Currency data object to be updated.

Returns

CurrencyData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyIDLabel" AssociatedControlID="uxCurrencyId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField  ID="uxCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyNameLabel" AssociatedControlID="uxCurrencyName"
CssClass="span-3 last" runat="server" Text=" Name:" />
        <ektronUI:TextField  ID="uxCurrencyName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyAlbhaLabel"
AssociatedControlID="uxCurrencyAlphaCode" CssClass="span-3 last" runat="server" Text="*
AlphaIsoCode:" />
        <ektronUI:TextField  ID="uxCurrencyAlphaCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyCultureCodeLabel"
AssociatedControlID="uxCurrencyCultureCode" CssClass="span-3 last" runat="server" Text="
Culture Code:" />
        <ektronUI:TextField  ID="uxCurrencyCultureCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
```

```
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CurrencyManager currencyManager = new CurrencyManager();
            int currencyId=int.Parse(uxCurrencyId.Text);
            CurrencyData currencyData = currencyManager.GetItem(currencyId);
            if (currencyData != null)
            {
                currencyData.AlphaIsoCode = uxCurrencyAlphaCode.Text;
                currencyData.CultureCode = uxCurrencyCultureCode.Text;
                currencyData.Name = uxCurrencyName.Text;
                currencyData = currencyManager.Update(currencyData);

                MessageUtilities.UpdateMessage(uxMessage, "Currency updated with Id " +
currencyData.Id, Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);


        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Data Classes

CurrencyCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `CurrencyCriteria()`

  ```
  public CurrencyCriteria()
  ```

- `CurrencyCriteria(Ektron.Cms.Commerce.CurrencyProperty, EkEnumeration.OrderByDirection)`

  ```
  public CurrencyCriteria(Ektron.Cms.Commerce.CurrencyProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

Accepted fields for the `CurrencyProperty` are:

- AlphaIsoCode
- CultureCode
- Enabled
- Id
- Name

CurrencyData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- AlphaIsoCode. Gets or sets the 3 letter ISO code of the currency.

  ```
  public string AlphaIsoCode { set; get; }
  ```

- `CultureCode`. Gets or sets the culture code associated with this currency. This is used for formatting currency values for this currency.

  ```
  public string CultureCode { set; get; }
  ```

- `CurrencySymbol`. Read only function that gets the symbol for this currency. Example: $

  ```
  public string CurrencySymbol { get; }
  ```

- `Enabled`. Gets or sets whether or not the currency is enabled for use.

  ```
  public bool Enabled { set; get; }
  ```

- `Id`. Gets or sets the Id of the currency. The Id is the ISO numeric code of the currency.

  ```
  public int Id { set; get; }
  ```

- `ISOCurrencySymbol`. Read only function that gets the ISO symbol for this currency. Example: USD

  ```
  public string ISOCurrencySymbol { get; }
  ```

- `Name`. Gets or sets the name of the currency.

  ```
  public string Name { set; get; }
  ```

CustomerManager

9.00 and higher

The class manipulates e-commerce customer details in CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.CustomerManager
```

Constructors

- `CustomerManager()`
- `CustomerManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add (billing address)

```
Add(Ektron.Cms.Commerce.CustomerData,Ektron.Cms.Commerce.AddressData)
```

Adds a new customer with a billing address. The Customer.Id property is populated with the new Customer's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Name
- * Password
- * Display Name
- First Name
- Last Name
- Email

Billing Address

- * Address Name
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID

Parameters

- `customerData`. Customer data to be added.
- `billingAddress`. Billing address of the customer. Also serves as shipping address.

Returns

Returns customer data with new customer's ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserNameLabel" AssociatedControlID="uxUserName"
CssClass="span-3 last"
            runat="server" Text="* UserName :" />
        <ektronUI:TextField ID="uxUserName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordLabel" AssociatedControlID="uxPassword"
CssClass="span-3 last"
            runat="server" Text="* Password :" />
        <asp:TextBox ID="uxPassword" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            TextMode="Password" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDisplayLabel" AssociatedControlID="uxDisplay"
CssClass="span-3 last"
            runat="server" Text="* DisplayName :" />
        <ektronUI:TextField ID="uxDisplay" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFirstNameLabel" AssociatedControlID="uxFirstName"
CssClass="span-3 last"
            runat="server" Text="FirstName :" />
        <ektronUI:TextField ID="uxFirstName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLastNameLabel" AssociatedControlID="uxLastName"
CssClass="span-3 last"
            runat="server" Text="LastName :" />
        <ektronUI:TextField ID="uxLastName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEmailLabel" AssociatedControlID="uxEmail" CssClass="span-3
last"
            runat="server" Text="Email :" />
        <ektronUI:TextField ID="uxEmail" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <b>Billing Address</b>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressNameLabel" AssociatedControlID="uxAddressName"
CssClass="span-3 last"
            runat="server" Text="* AddressName :" />
        <ektronUI:TextField ID="uxAddressName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressLine1Label" AssociatedControlID="uxAddressLine1"
CssClass="span-3 last"
            runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxAddressLine1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCityLabel" AssociatedControlID="uxCity" CssClass="span-3
last"
```

```
            runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLagel" AssociatedControlID="uxRegionId"
CssClass="span-3 last"
            runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPostalCodeLabel" AssociatedControlID="uxPostalCode"
CssClass="span-3 last"
            runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxPostalCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIdLabel" AssociatedControlID="uxCountryId"
CssClass="span-3 last"
            runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            string password = uxPassword.Text;
            string display = uxDisplay.Text;
            long regionId;
            long.TryParse(uxRegionId.Text, out regionId);
            int countryId;
            int.TryParse(uxCountryId.Text, out countryId);
            string emailAddress = uxEmail.Text;
            if (!string.IsNullOrEmpty(emailAddress))
            {
                try
                {
                    System.Net.Mail.MailAddress addr = new System.Net.Mail.MailAddress
(emailAddress);
                }
                catch
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Email
Address.", Message.DisplayModes.Error);
                    return;
                }
            }

            if (password != "" && display != "" && regionId > 0 && countryId > 0)
            {
                CustomerManager customerManager = new CustomerManager();

                //Customer Information
                CustomerData customerData = new CustomerData();
                customerData.UserName = uxUserName.Text; ;
                customerData.Password = password;
                customerData.FirstName = uxFirstName.Text;
                customerData.LastName = uxLastName.Text;
                customerData.EmailAddress = emailAddress;
                customerData.DisplayName = display;
                customerData.IsMembershipUser = true;
                //BillingAddress Information
                AddressData billingAddress = new AddressData();
                billingAddress.Name = uxAddressName.Text;
                billingAddress.AddressLine1 = uxAddressLine1.Text;
                billingAddress.City = uxCity.Text;
                billingAddress.Region = new RegionData { Id = regionId };
                billingAddress.PostalCode = uxPostalCode.Text;
                billingAddress.Country = new CountryData { Id = countryId };

                long customerId = customerManager.Add(customerData, billingAddress).Id;
                MessageUtilities.UpdateMessage(uxMessage, "Customer Added with Id: " +
customerId.ToString(), Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields", Message.DisplayModes.Error);
            }
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Add (customer ID)

```
Add(Ektron.Cms.Commerce.CustomerData)
```

Adds a new customer. The Customer.Id property is populated with the new Customer's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Name
- * Password
- * Display Name
- First Name
- Last Name
- Email

Parameters

- `customerData`. Customer data to be added.

Returns

Returns customer data with new customer's ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserNameLabel" AssociatedControlID="uxUserName"
CssClass="span-3 last"
            runat="server" Text="* UserName :" />
        <ektronUI:TextField ID="uxUserName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxPasswordLabel" AssociatedControlID="uxPassword"
CssClass="span-3 last"
            runat="server" Text="* Password :" />
        <asp:TextBox ID="uxPassword" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            TextMode="Password" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDisplayLabel" AssociatedControlID="uxDisplay"
CssClass="span-3 last"
            runat="server" Text="* DisplayName :" />
        <ektronUI:TextField ID="uxDisplay" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFirstNameLabel" AssociatedControlID="uxFirstName"
CssClass="span-3 last"
            runat="server" Text="FirstName :" />
        <ektronUI:TextField ID="uxFirstName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLastNameLabel" AssociatedControlID="uxLastName"
CssClass="span-3 last"
            runat="server" Text="LastName :" />
        <ektronUI:TextField ID="uxLastName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Label ID="uxEmailLabel" AssociatedControlID="uxEmail" CssClass="span-3
last"
            runat="server" Text="Email :" />
        <ektronUI:TextField ID="uxEmail" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            string password = uxPassword.Text;
            string display = uxDisplay.Text;
            string emailAddress = uxEmail.Text;
            if (!string.IsNullOrEmpty(emailAddress))
            {
                try
                {
                    System.Net.Mail.MailAddress addr = new System.Net.Mail.MailAddress
(emailAddress);
                }
                catch
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Email
Address.", Message.DisplayModes.Error);
                    return;
                }
            }

            if (password != "" && display != "")
            {
                CustomerManager customerManager = new CustomerManager();
                CustomerData customerData = new CustomerData();
                customerData.UserName = uxUserName.Text;
                customerData.Password = password;
                customerData.FirstName = uxFirstName.Text;
                customerData.LastName = uxLastName.Text;
                customerData.EmailAddress = emailAddress;
                customerData.DisplayName = display;
                customerData.IsMembershipUser = true;
                long customerId = customerManager.Add(customerData).Id;
                MessageUtilities.UpdateMessage(uxMessage, "Customer Added with Id: " +
customerId.ToString(), Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
```

```
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Add (shipping address)

```
Add
(Ektron.Cms.Commerce.CustomerData,Ektron.Cms.Commerce.AddressData,Ektron.Cms.Commerce.Ad
dressData)
```

Adds a new customer with separate billing and shipping addresses. The Customer.Id property is populated with the new Customer's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Name
- * Password
- * Display Name
- First Name
- Last Name
- Email

Billing Address

- * Address Name
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID

Shipping Address

- * Address Name
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID

Parameters

- `customerData`. Customer data to be added.
- `billingAddress`. Billing address of the customer.
- `shippingAddress`. Shipping address of the customer.

Returns

Returns customer data with new customer's ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserNameLabel" AssociatedControlID="uxUserName"
CssClass="span-3 last"
            runat="server" Text="* UserName :" />
        <ektronUI:TextField ID="uxUserName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordLabel" AssociatedControlID="uxPassword"
CssClass="span-3 last"
            runat="server" Text="* Password :" />
        <asp:TextBox ID="uxPassword" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            TextMode="Password" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDisplayLabel" AssociatedControlID="uxDisplay"
CssClass="span-3 last"
            runat="server" Text="* DisplayName :" />
        <ektronUI:TextField ID="uxDisplay" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFirstNameLabel" AssociatedControlID="uxFirstName"
CssClass="span-3 last"
            runat="server" Text="FirstName :" />
        <ektronUI:TextField ID="uxFirstName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLastNameLabel" AssociatedControlID="uxLastName"
CssClass="span-3 last"
            runat="server" Text="LastName :" />
        <ektronUI:TextField ID="uxLastName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEmailLabel" AssociatedControlID="uxEmail" CssClass="span-3
last"
            runat="server" Text="Email :" />
        <ektronUI:TextField ID="uxEmail" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <b>Billing Address</b>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxBillingAddressNameLabel"
AssociatedControlID="uxBillingAddressName" CssClass="span-3 last"
            runat="server" Text="* AddressName :" />
        <ektronUI:TextField ID="uxBillingAddressName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxBillingAddressLine1Label"
AssociatedControlID="uxBillingAddressLine1" CssClass="span-3 last"
            runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxBillingAddressLine1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxBillingCityLabel" AssociatedControlID="uxBillingCity"
CssClass="span-3 last"
            runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxBillingCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxBillingRegionIdLagel"
AssociatedControlID="uxBillingRegionId" CssClass="span-3 last"
            runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxBillingRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxBillingPostalCodeLabel"
AssociatedControlID="uxBillingPostalCode" CssClass="span-3 last"
            runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxBillingPostalCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxBillingCountryIdLabel"
AssociatedControlID="uxBillingCountryId" CssClass="span-3 last"
            runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxBillingCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <b>Shipping Address</b>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingAddressNameLabel"
AssociatedControlID="uxShippingAddressName" CssClass="span-3 last"
            runat="server" Text="* AddressName :" />
        <ektronUI:TextField ID="uxShippingAddressName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingAddressLine1Label"
AssociatedControlID="uxShippingAddressLine1" CssClass="span-3 last"
            runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxShippingAddressLine1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingCityLabel" AssociatedControlID="uxShippingCity"
CssClass="span-3 last"
            runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxShippingCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingRegionIdLabel"
AssociatedControlID="uxShippingRegionId" CssClass="span-3 last"
            runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxShippingRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingPostalCodeLabel"
AssociatedControlID="uxShippingPostalCode" CssClass="span-3 last"
            runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxShippingPostalCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingCountryIdLabel"
AssociatedControlID="uxShippingCountryId" CssClass="span-3 last"
            runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxShippingCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            string password = uxPassword.Text;
            string display = uxDisplay.Text;
            long regionId;
            long.TryParse(uxRegionId.Text, out regionId);
            int countryId;
            int.TryParse(uxCountryId.Text, out countryId);
            string emailAddress = uxEmail.Text;
            if (!string.IsNullOrEmpty(emailAddress))
            {
                try
                {
                    System.Net.Mail.MailAddress addr = new System.Net.Mail.MailAddress
(emailAddress);
                }
                catch
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Email
Address.", Message.DisplayModes.Error);
                    return;
                }
            }

            if (password != "" && display != "" && regionId > 0 && countryId > 0)
            {
                CustomerManager customerManager = new CustomerManager();

                //Customer Information
                CustomerData customerData = new CustomerData();
                customerData.UserName = uxUserName.Text; ;
                customerData.Password = password;
                customerData.FirstName = uxFirstName.Text;
                customerData.LastName = uxLastName.Text;
                customerData.EmailAddress = emailAddress;
                customerData.DisplayName = display;
                customerData.IsMembershipUser = true;
                //BillingAddress Information
                AddressData billingAddress = new AddressData();
                billingAddress.Name = uxAddressName.Text;
                billingAddress.AddressLine1 = uxAddressLine1.Text;
                billingAddress.City = uxCity.Text;
                billingAddress.Region = new RegionData { Id = regionId };
                billingAddress.PostalCode = uxPostalCode.Text;
                billingAddress.Country = new CountryData { Id = countryId };
```

```
                long customerId = customerManager.Add(customerData, billingAddress).Id;
                MessageUtilities.UpdateMessage(uxMessage, "Customer Added with Id: " +
customerId.ToString(), Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a customer from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Customer ID

Parameters

- `customerId`. ID of the customer to be deleted.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCustomerIdLabel" AssociatedControlID="uxCustomerId"
CssClass="span-3 last"
            runat="server" Text="* CustomerId :" />
        <ektronUI:TextField ID="uxCustomerId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
```

```
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CustomerManager customerManager = new CustomerManager();
            long customerId;
            long.TryParse(uxCustomerId.Text, out customerId);
            if (customerId > 0)
            {
                CustomerData customerData = customerManager.GetItem(customerId);
                if (customerData != null)
                {
                    customerManager.Delete(customerId);
                    MessageUtilities.UpdateMessage(uxMessage, "Customer with Id " +
customerId.ToString() + " has been Deleted.", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
Customer Id", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Customer
Id", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a customer.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Customer ID

Parameters

- `customerId`. ID of the customer to be retrieved.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCustomerIdLabel" AssociatedControlID="uxCustomerId"
CssClass="span-3 last"
            runat="server" Text="* CustomerId :" />
        <ektronUI:TextField ID="uxCustomerId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFirstName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLastName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDisplayName" runat="server"></asp:Literal>
```

```
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxEmail" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CustomerManager customerManager = new CustomerManager();
            long customerId;
            long.TryParse(uxCustomerId.Text, out customerId);
            if (customerId > 0)
            {
                CustomerData customerData = customerManager.GetItem(customerId);
                if (customerData != null)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Customer Details for
Customer Id : " + customerId.ToString() + " are below", Message.DisplayModes.Success);
                    uxUserName.Text = "UserName : " + customerData.UserName;
                    uxFirstName.Text = "FirstName : " + customerData.FirstName;
                    uxLastName.Text = "LastName : " + customerData.LastName;
                    uxDisplayName.Text = "DisplayName : " + customerData.DisplayName;
                    uxEmail.Text = "Email : " + customerData.EmailAddress;
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
Customer Id", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Customer
Id", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
```

```
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.CustomerCriteria)
```

Returns a list of customers based upon supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Customer Property
- * Object Value

Parameters

- `criteria`. Criteria by which to retrieve customer objects.

Returns

List of CustomerData.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCustomerPropertyLabel"
AssociatedControlID="uxCustomerProperty" CssClass="span-4 last"
            runat="server" Text="* CustomerProperty :" />
        <asp:DropDownList ID="uxCustomerProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>UserName</asp:ListItem>
            <asp:ListItem>DisplayName</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last"
            runat="server" Text="* ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCustomerlist" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Name
                    </th>
                    <th>
                        Display Name
                    </th>
                    <th>
                        Email
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("DisplayName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("EmailAddress")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object objectValue = uxObjectValue.Text;
            string customerProperty = uxCustomerProperty.SelectedItem.Text;
            CustomerManager customerManager = new CustomerManager();
            CustomerCriteria criteria = new CustomerCriteria(CustomerProperty.UserName,
EkEnumeration.OrderByDirection.Ascending);
            if (customerProperty == "Id")
            {
                criteria.AddFilter(CustomerProperty.Id, CriteriaFilterOperator.EqualTo,
objectValue);
            }
            else if (customerProperty == "UserName")
            {
                criteria.AddFilter(CustomerProperty.UserName,
CriteriaFilterOperator.EqualTo, objectValue);
            }
            else
            {
                criteria.AddFilter(CustomerProperty.DisplayName,
CriteriaFilterOperator.EqualTo, objectValue);
            }

            List<CustomerData> customerDataList = customerManager.GetList(criteria);
            uxCustomerlist.Visible = true;
            uxCustomerlist.DataSource = customerDataList;
            uxCustomerlist.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IsUserNameExists

```
IsUserNameExists(System.String)
```

Returns whether the given UserName is available to add new user.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Name

Parameters

- `userName`. Name of the User.

Returns

Returns true if UserName is available to add a new user; otherwise false.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserNameLabel" AssociatedControlID="uxUserName"
CssClass="span-3 last"
            runat="server" Text="* UserName :" />
        <ektronUI:TextField ID="uxUserName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="IsUserNameExists"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
```

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CustomerManager customerManager = new CustomerManager();
            string userName = uxUserName.Text;
            if (userName != "")
            {
                bool isAvailable = customerManager.IsUserNameExists(userName);
                if (isAvailable == true)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "UserName " + userName + "
is exists in the system", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "UserName " + userName + "
is not exists in the system", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid User
Name.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

SetBillingAddress

```
SetBillingAddress(System.Int64,System.Int64)
```

Sets a customers billing address.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Customer ID
- * Address ID

Parameters

- `customerId`. ID of the customer.
- `addressId`. ID of the address to be set as billing address to the customer.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCustomerIdLabel" AssociatedControlID="uxCustomerId"
CssClass="span-3 last"
            runat="server" Text="* CustomerId :" />
        <ektronUI:TextField ID="uxCustomerId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressIdLabel" AssociatedControlID="uxAddressId"
CssClass="span-3 last"
            runat="server" Text="* AddressId :" />
        <ektronUI:TextField ID="uxAddressId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="SetBillingAddress"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CustomerManager customerManager = new CustomerManager();
            long customerId;
```

```
            long.TryParse(uxCustomerId.Text, out customerId);
            long addressId;
            long.TryParse(uxAddressId.Text, out addressId);
            if (customerId > 0 && addressId > 0)
            {
                CustomerData customerData = customerManager.GetItem(customerId);
                if (customerData != null)
                {
                    customerManager.SetBillingAddress(customerId, addressId);
                    MessageUtilities.UpdateMessage(uxMessage, "Billing Address Id " +
addressId.ToString() + " is updated to Customer Id " + customerId.ToString(),
Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
Customer Id", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

SetShippingAddress

```
SetShippingAddress(System.Int64,System.Int64)
```

Sets a customers shipping address.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Customer ID
- * Address ID

Parameters

- `customerId`. ID of the customer.
- `addressId`. ID of the address to be set as shipping address to the customer.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCustomerIdLabel" AssociatedControlID="uxCustomerId"
CssClass="span-3 last"
            runat="server" Text="* CustomerId :" />
        <ektronUI:TextField ID="uxCustomerId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressIdLabel" AssociatedControlID="uxAddressId"
CssClass="span-3 last"
            runat="server" Text="* AddressId :" />
        <ektronUI:TextField ID="uxAddressId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="SetShippingAddress"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CustomerManager customerManager = new CustomerManager();
            long customerId;
            long.TryParse(uxCustomerId.Text, out customerId);
            long addressId;
            long.TryParse(uxAddressId.Text, out addressId);
            if (customerId > 0 && addressId > 0)
```

```
            {
                CustomerData customerData = customerManager.GetItem(customerId);
                if (customerData != null)
                {
                    customerManager.SetShippingAddress(customerId, addressId);
                    MessageUtilities.UpdateMessage(uxMessage, "Shipping Address Id " +
addressId.ToString() + " is updated to Customer Id " + customerId.ToString(),
Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
Customer Id", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.CustomerData)
```

Updates an existing customer.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Customer ID
- User Name
- Password
- Display Name
- First Name
- Last Name
- Email

Parameters

- `customerData`. Customer data to be updated.

Returns

Returns customer data.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCustomerIdLabel" AssociatedControlID="uxCustomerId"
CssClass="span-3 last"
            runat="server" Text="* CustomerId :" />
        <ektronUI:TextField ID="uxCustomerId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserNameLabel" AssociatedControlID="uxUserName"
CssClass="span-3 last"
            runat="server" Text=" UserName :" />
        <ektronUI:TextField ID="uxUserName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordLabel" AssociatedControlID="uxPassword"
CssClass="span-3 last"
            runat="server" Text=" Password :" />
        <asp:TextBox ID="uxPassword" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            TextMode="Password" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDisplayLabel" AssociatedControlID="uxDisplay"
CssClass="span-3 last"
            runat="server" Text=" DisplayName :" />
        <ektronUI:TextField ID="uxDisplay" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFirstNameLabel" AssociatedControlID="uxFirstName"
CssClass="span-3 last"
            runat="server" Text=" FirstName :" />
        <ektronUI:TextField ID="uxFirstName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLastNameLabel" AssociatedControlID="uxLastName"
CssClass="span-3 last"
            runat="server" Text=" LastName :" />
        <ektronUI:TextField ID="uxLastName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxEmailLabel" AssociatedControlID="uxEmail" CssClass="span-3
last"
            runat="server" Text=" Email :" />
        <ektronUI:TextField ID="uxEmail" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long customerId;
            long.TryParse(uxCustomerId.Text, out customerId);
            string emailAddress = uxEmail.Text;
            if (!string.IsNullOrEmpty(emailAddress))
            {
                try
                {
                    System.Net.Mail.MailAddress addr = new System.Net.Mail.MailAddress
(emailAddress);
                }
                catch
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Email
Address.", Message.DisplayModes.Error);
                    return;
                }
```

```
            }

            if (customerId > 0)
            {
                CustomerManager customerManager = new CustomerManager();
                CustomerData customerData = customerManager.GetItem(customerId);
                if (customerData != null)
                {
                    customerData.UserName = uxUserName.Text != "" ? uxUserName.Text :
customerData.UserName;
                    customerData.Password = uxPassword.Text != "" ? uxPassword.Text :
customerData.Password;
                    customerData.FirstName = uxFirstName.Text != "" ? uxFirstName.Text :
customerData.FirstName;
                    customerData.LastName = uxLastName.Text != "" ? uxLastName.Text :
customerData.LastName;
                    customerData.EmailAddress = emailAddress != "" ? emailAddress :
customerData.EmailAddress;
                    customerData.DisplayName = uxDisplay.Text != "" ? uxDisplay.Text :
customerData.DisplayName;
                    customerManager.Update(customerData);
                    MessageUtilities.UpdateMessage(uxMessage, "Customer Updated for Id:
" + customerId.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
Customer Id", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Customer
Id", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

CustomerCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `CustomerCriteria()`

```
public CustomerCriteria()
```

- `CustomerCriteria(Ektron.Cms.Commerce.CustomerProperty, EkEnumeration.OrderByDirection))`

```
public CustomerCriteria(Ektron.Cms.Commerce.CustomerProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `CustomerProperty` are:

- BillingAddress
- DateCreated
- DisplayName
- EmailAddress
- FirstName
- Id
- IsDeleted
- IsMember
- LanguageId
- LastName
- ShippingAddressId
- TotalOrders
- TotalOrderValue
- UserName

CustomerData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `AverageOrderValue`. Gets the average order placed by this customer.

```
public decimal AverageOrderValue { get; }
```

- `BillingAddressId`. Gets or sets the Address Id of the billing address for the customer.

```
public long BillingAddressId { set; get; }
```

- `DateCreated`

```
public System.DateTime DateCreated { set; get; }
```

- `DisplayName`. Gets or sets the display name of the customer.

```
public string DisplayName { set; get; }
```

- `EmailAddress`. Gets or sets the email address of the customer.

```
public string EmailAddress { set; get; }
```

- `FirstName`. Gets or sets the first name of the customer.

```
public string FirstName { set; get; }
```

- `Id`. Gets or sets the Id of the customer.

```
public long Id { set; get; }
```

- `IsDeleted`. Gets the IsDeleted status of the customer.

```
public bool IsDeleted { set; get; }
```

- `IsMembershipUser`. Gets or sets the IsMembershipUser status of the customer.

```
public bool IsMembershipUser { set; get; }
```

- `LanguageId`. Gets or sets the integer ID of the language for the UserData object. Returns the integer ID of the language.

```
public int LanguageId { set; get; }
```

- `LastName`. Gets or sets the last of the customer.

```
public string LastName { set; get; }
```

- `Password`. Gets or sets the password for the customer.

```
public string Password { set; get; }
```

- `ShippingAddressId`. Gets or sets Address Id of the shipping address for this customer.

```
public long ShippingAddressId { set; get; }
```

- `TotalOrders`. A read-only property, it gets the total number of orders placed by the customer.

```
public int TotalOrders { set; get; }
```

- `TotalOrderValue`. A read-only property, it gets the total value of orders placed by the customer.

```
public decimal TotalOrderValue { set; get; }
```

- `UserId`. Gets or sets the Id of the Cms User object associated with this customer.

```
public long UserId { set; get; }
```

- `UserName`. Gets or sets the username for the customer.

```
public string UserName { set; get; }
```

ExchangeRateManager

9.00 and higher

The class manages commerce exchange rates in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.ExchangeRateManager
```

Constructors

- `ExchangeRateManager()`
- `ExchangeRateManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 553
- *GetCurrentExchangeRate* on page 555
- *GetCurrentExchangeRate (ID)* on page 557
- *GetCurrentList* on page 559
- *GetList* on page 562
- *Update* on page 565

Add

```
Add(Ektron.Cms.Commerce.ExchangeRateData)
```

Adds a new exchange rate to the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Base Currency ID
- * Exchange Currency ID
- Rate
- * Effective Date

Parameters

- `exchangeRateData`. Exchange rate data to be added.

Returns

ExchangeRateData object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxBaseCurrencyIdLabel"
AssociatedControlID="uxBaseCurrencyId" CssClass="span-3 last" runat="server" Text="*
BaseCurrency Id:" />
        <ektronUI:TextField  ID="uxBaseCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExchangeCurrencyIdLabel"
AssociatedControlID="uxExchangeCurrencyId" CssClass="span-3 last" runat="server" Text="*
ExchangeCurrency Id:" />
        <ektronUI:TextField  ID="uxExchangeCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyRateLabel" AssociatedControlID="uxCurrencyRate"
CssClass="span-3 last" runat="server" Text=" Rate:" />
        <ektronUI:TextField  ID="uxCurrencyRate" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEffectiveDateLabel" AssociatedControlID="uxEffectiveDate"
CssClass="span-3 last" runat="server" Text="* EffectiveDate:" />
        <ektronUI:DatePicker Rows="3" ID="uxEffectiveDate" CssClass="span-4"
runat="server"
            Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxEffectiveDateTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
```

```
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEffectiveDatePeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            ExchangeRateManager exchangeRateManager = new ExchangeRateManager();

            string strEffectiveDate = uxEffectiveDate.Text + " " +
uxEffectiveDateTime.SelectedItem.Text + ":00:00 " +
uxEffectiveDatePeriod.SelectedItem.Text;
            DateTime EffectiveDate = DateTime.Parse(strEffectiveDate);

            ExchangeRateData exchangeRateData = new ExchangeRateData()
            {
                BaseCurrencyId = int.Parse(uxBaseCurrencyId.Text),
```

```
                ExchangeCurrencyId = int.Parse(uxExchangeCurrencyId.Text),
                Rate = decimal.Parse(uxCurrencyRate.Text),
                EffectiveDate = EffectiveDate,
            };

            exchangeRateManager.Add(exchangeRateData);

            MessageUtilities.UpdateMessage(uxMessage, "ExchangeRate is Added." ,
Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(Ektron.Cms.Commerce.ExchangeRateData)
```

Deletes an exchange rate from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Base Currency ID
- * Exchange Currency ID
- * Effective Date

Parameters

- `exchangeRateData`. ExchangeRateData to be deleted.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxBaseCurrencyIdLabel"
AssociatedControlID="uxBaseCurrencyId" CssClass="span-3 last" runat="server" Text="*
BaseCurrency Id:" />
        <ektronUI:TextField  ID="uxBaseCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxExchangeCurrencyIdLabel"
AssociatedControlID="uxExchangeCurrencyId" CssClass="span-3 last" runat="server" Text="*
ExchangeCurrency Id:" />
        <ektronUI:TextField  ID="uxExchangeCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEffectiveDateLabel" AssociatedControlID="uxEffectiveDate"
CssClass="span-3 last" runat="server" Text="* EffectiveDate:" />
        <ektronUI:DatePicker Rows="3" ID="uxEffectiveDate" CssClass="span-4"
runat="server"
            Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxEffectiveDateTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEffectiveDatePeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            ExchangeRateManager exchangeRateManager = new ExchangeRateManager();
            string strEffectiveDate = uxEffectiveDate.Text + " " +
uxEffectiveDateTime.SelectedItem.Text + ":00:00 " +
uxEffectivePeriod.SelectedItem.Text;
            DateTime EffectiveDate = DateTime.Parse(strEffectiveDate);

            ExchangeRateData exchangeRateData = new ExchangeRateData()
            {
                BaseCurrencyId = int.Parse(uxBaseCurrencyId.Text),
                ExchangeCurrencyId = int.Parse(uxExchangeCurrencyId.Text),

                EffectiveDate = EffectiveDate,
            };

            exchangeRateManager.Delete(exchangeRateData);

            MessageUtilities.UpdateMessage(uxMessage, "ExchangeRate is Deleted.",
Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetCurrentExchangeRate

```
GetCurrentExchangeRate()
```

Retrieves the current exchange rate.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Returns

ExchangeRateData object retrieved.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetCurrentExchangeRate"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxBaseCurrencyId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxExchangeId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxRate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxEffectiveDate" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            ExchangeRateManager exchangeRateManager = new ExchangeRateManager();

            ExchangeRateData exchangeRateData =
exchangeRateManager.GetCurrentExchangeRate();

            if (exchangeRateData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for ExchangeRate with
```

```
Exchange CurrencyId " + exchangeRateData.ExchangeCurrencyId.ToString() + " are below",
Message.DisplayModes.Success);
                uxBaseCurrencyId.Text = "Base CurrencyId : " +
exchangeRateData.BaseCurrencyId;
                uxExchangeId.Text = "Exchange CurrencyId : " +
exchangeRateData.ExchangeCurrencyId;
                uxRate.Text = "Rate : " + exchangeRateData.Rate;
                uxEffectiveDate.Text = "Effective Date : " +
exchangeRateData.EffectiveDate;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Current exchange data does
not exist. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetCurrentExchangeRate (ID)

```
GetCurrentExchangeRate(System.Int32)
```

Retrieves the current exchange rate by exchange currency ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Exchange Currency ID

Parameters

- `exchangeCurrencyId`. ID of exchange rate to retrieve.

Returns

ExchangeRateData object retrieved.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxExchangeCurrencyIDLabel"
AssociatedControlID="uxExchangeCurrencyId" CssClass="span-3 last" runat="server"
Text="*Exchange CurrencyId:" />
        <ektronUI:TextField ID="uxExchangeCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetCurrentExchangeRate"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxBaseCurrencyId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxExchangeId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxRate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxEffectiveDate" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            ExchangeRateManager exchangeRateManager = new ExchangeRateManager();
            int exchangeRateId=int.Parse(uxExchangeCurrencyId.Text);
            ExchangeRateData exchangeRateData =
exchangeRateManager.GetCurrentExchangeRate(exchangeRateId);
```

```
            if (exchangeRateData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for ExchangeRate with
Exchange CurrencyId " + exchangeRateData.ExchangeCurrencyId.ToString() + " are below",
Message.DisplayModes.Success);
                uxBaseCurrencyId.Text = "Base CurrencyId : " +
exchangeRateData.BaseCurrencyId;
                uxExchangeId.Text = "Exchange CurrencyId : " +
exchangeRateData.ExchangeCurrencyId;
                uxRate.Text = "Rate : " + exchangeRateData.Rate;
                uxEffectiveDate.Text = "Effective Date : " +
exchangeRateData.EffectiveDate;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "ExchangeRate " +
exchangeRateId + " does not exist. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetCurrentList

```
GetCurrentList(Ektron.Cms.Commerce.ExchangeRateCriteria)
```

Retrieves a list of current exchange rates.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Exchange Rate Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter exchange rates being retrieved.

Returns

List of exchange rates meeting criteria filters.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxExchangeRatePropertyLabel"
AssociatedControlID="uxExchangeRateProperty" CssClass="span-4 last" runat="server"
Text="ExchangeRate Property:" />
        <asp:DropDownList ID="uxExchangeRateProperty" runat="server">
            <asp:ListItem>BaseCurrencyId</asp:ListItem>
            <asp:ListItem>ExchangeCurrencyId</asp:ListItem>
            <asp:ListItem>Rate </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetCurrentList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxExchangeRateListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Base CurrencyId
                    </th>
                    <th>
                       Exchange CurrencyId
                    </th>
                    <th>
                        Rate
                    </th>
                    <th>
                       Effective Date
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
```

```
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("BaseCurrencyId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ExchangeCurrencyId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Rate")%>
            </td>
            <td class="devsite-method">
                <%# Eval("EffectiveDate")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            ExchangeRateCriteria criteria = new ExchangeRateCriteria();

            if (uxExchangeRateProperty.SelectedItem.Text == "BaseCurrencyId")
            {
                Objectvalue = int.Parse(uxObjectValue.Text);
                criteria.AddFilter(ExchangeRateProperty.BaseCurrencyId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxExchangeRateProperty.SelectedItem.Text == "ExchangeCurrencyId")
            {
                Objectvalue = int.Parse(uxObjectValue.Text);
                criteria.AddFilter(ExchangeRateProperty.ExchangeCurrencyId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
```

```
                Objectvalue =Decimal.Parse(uxObjectValue.Text);
                criteria.AddFilter(ExchangeRateProperty.Rate,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            ExchangeRateManager exchangeRateManger = new ExchangeRateManager();
            List<ExchangeRateData> exchangeRateList = exchangeRateManger.GetCurrentList
(criteria);

            uxExchangeRateListView.Visible = true;
            uxExchangeRateListView.DataSource = exchangeRateList;
            uxExchangeRateListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.ExchangeRateCriteria)
```

Retrieves a list of exchange rates.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Exchange Rate Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter exchange rates being retrieved.

Returns

List of exchange rates meeting criteria filters.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxExchangeRatePropertyLabel"
AssociatedControlID="uxExchangeRateProperty" CssClass="span-4 last" runat="server"
Text="ExchangeRate Property:" />
        <asp:DropDownList ID="uxExchangeRateProperty" runat="server">
            <asp:ListItem>BaseCurrencyId</asp:ListItem>
            <asp:ListItem>ExchangeCurrencyId</asp:ListItem>
            <asp:ListItem>Rate </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxExchangeRateListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Base CurrencyId
                    </th>
                    <th>
                       Exchange CurrencyId
                    </th>
                    <th>
                        Rate
                    </th>
                    <th>
                       Effective Date
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
```

```
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("BaseCurrencyId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ExchangeCurrencyId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Rate")%>
            </td>
            <td class="devsite-method">
                <%# Eval("EffectiveDate")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            ExchangeRateCriteria criteria = new ExchangeRateCriteria();

            if (uxExchangeRateProperty.SelectedItem.Text == "BaseCurrencyId")
            {
                Objectvalue = int.Parse(uxObjectValue.Text);
                criteria.AddFilter(ExchangeRateProperty.BaseCurrencyId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxExchangeRateProperty.SelectedItem.Text == "ExchangeCurrencyId")
            {
                Objectvalue = int.Parse(uxObjectValue.Text);
                criteria.AddFilter(ExchangeRateProperty.ExchangeCurrencyId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue =Decimal.Parse(uxObjectValue.Text);
                criteria.AddFilter(ExchangeRateProperty.Rate,
```

```
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            ExchangeRateManager exchangeRateManger = new ExchangeRateManager();
            List<ExchangeRateData> exchangeRateList = exchangeRateManger.GetList
(criteria);

            uxExchangeRateListView.Visible = true;
            uxExchangeRateListView.DataSource = exchangeRateList;
            uxExchangeRateListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.ExchangeRateData)
```

Updates an existing exchange rate in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Base Currency ID
- * Exchange Currency ID
- * Effective Date
- Rate

Parameters

- `exchangeRateData`. ExchangeRateData to be updated.

Returns

Exchange rate data object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxBaseCurrencyIdLabel"
AssociatedControlID="uxBaseCurrencyId" CssClass="span-3 last" runat="server" Text="*
BaseCurrency Id:" />
        <ektronUI:TextField  ID="uxBaseCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExchangeCurrencyIdLabel"
AssociatedControlID="uxExchangeCurrencyId" CssClass="span-3 last" runat="server" Text="*
ExchangeCurrency Id:" />
        <ektronUI:TextField  ID="uxExchangeCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEffectiveDateLabel" AssociatedControlID="uxEffectiveDate"
CssClass="span-3 last" runat="server" Text="* EffectiveDate:" />
        <ektronUI:DatePicker Rows="3" ID="uxEffectiveDate" CssClass="span-4"
runat="server"
            Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxEffectiveDateTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEffectiveDatePeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyRateLabel" AssociatedControlID="uxCurrencyRate"
CssClass="span-3 last" runat="server" Text=" Rate:" />
        <ektronUI:TextField  ID="uxCurrencyRate" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            ExchangeRateManager exchangeRateManager = new ExchangeRateManager();
            string strEffectiveDate = uxEffectiveDate.Text + " " +
uxEffectiveDateTime.SelectedItem.Text + ":00:00 " +
uxEffectiveDatePeriod.SelectedItem.Text;
            DateTime EffectiveDate = DateTime.Parse(strEffectiveDate);

            ExchangeRateData exchangeRateData = new ExchangeRateData()
            {
                BaseCurrencyId = int.Parse(uxBaseCurrencyId.Text),
                ExchangeCurrencyId = int.Parse(uxExchangeCurrencyId.Text),
                Rate = decimal.Parse(uxCurrencyRate.Text),
                EffectiveDate = EffectiveDate,
            };

            exchangeRateManager.Update(exchangeRateData);

            MessageUtilities.UpdateMessage(uxMessage, "ExchangeRate is Updated.",
Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Data Classes

ExchangeRateCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `ExchangeRateCriteria()`

  ```
  public ExchangeRateCriteria()
  ```

- `ExchangeRateCriteria(Ektron.Cms.Commerce.ExchangeRateProperty, EkEnumeration.OrderByDirection)`

  ```
  public ExchangeRateCriteria(Ektron.Cms.Commerce.ExchangeRateProperty orderByField,
  EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `ExchangeRateProperty` are:

  - BaseCurrencyId
  - EffectiveDate
  - ExchangeCurrencyId
  - Rate

ExchangeRateData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `BaseCurrency`. Gets or sets the Currency Id for the base currency. This is the currency that is being used as the basis for the exchange rate. If you are converting from USD to AUD, USD is the base currency.

  ```
  public int BaseCurrencyId { set; get; }
  ```

- `ConvertPrice(decimal)`. Converts a pric ein the base currency to a price in the exchange currency using the rate defined.

  ```
  public decimal ConvertPrice(decimal basePrice)
  ```

- `EffectiveDate`. Gets or sets the date and time in which this rate becomes effective.

  ```
  public System.DateTime? EffectiveDate { set; get; }
  ```

- `ExchangeCurrencyId`. Gets or sets the Currency Id for the exchange currency. This is the currency that the currency amount is being converted to. If you are converting from USD to AUD, AUD is the exchange currency.

  ```
  public int ExchangeCurrencyId { set; get; }
  ```

- `ExchangeRateData`

  ```
  public ExchangeRateData()
  ```

- `ExchangeRateData(int, int, decimal, System.DateTime?)`

  ```
  public ExchangeRateData(int baseCurrencyId,
    int exchangeCurrencyId, decimal rate,
    System.DateTime? effectiveDate)
  ```

- `ExchangeCurrencyId`. Gets or sets the Currency Id for the exchange currency. This is the currency that the currency amount is being converted to. If you are converting from USD to AUD, AUD is the exchange currency.

```
public int ExchangeCurrencyId { set; get; }
```

- `Rate`. Gets or sets the actual exchange rate between to currencies. This is the rate applied to base currency to convert it to exchange currency.

```
public decimal Rate { set; get; }
```

- `Validate()`. Validates the ExchangeRate data object for saving. Returns a ValidationResult object with results.

```
public ValidationResults Validate()
```

InventoryManager

9.00 and higher

The class manages inventory in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.InventoryManager
```

Constructors

- `InventoryManager()`
- `InventoryManager(ApiAccessMode)`.

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Commerce.InventoryData)
```

Adds the inventory data to catalog entry item.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry ID
- * Units in Stock
- * Reorder Level
- Disable Inventory

Parameters

- `inventoryData`. Inventory data object to be added.

Returns

InventoryData object.

.aspx code snippet

```
<ol class="formFields">

    <li class="clearfix">
        <ektronUI:Label ID="uxEntryidLabel" AssociatedControlID="uxEntryId"
CssClass="span-4 last" runat="server" Text="* Entry Id:" />
        <ektronUI:TextField  ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQuantityLabel" AssociatedControlID="uxUnitsInStock"
CssClass="span-4 last" runat="server" Text="* Units In Stock:" />
        <ektronUI:TextField  ID="uxUnitsInStock" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxReorderLevelLabel" AssociatedControlID="uxReorderLevel"
CssClass="span-4 last" runat="server" Text="* Reorder Level:" />
        <ektronUI:TextField  ID="uxReorderLevel" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxInventoryDissabledLable"
AssociatedControlID="uxInventoryDissabled" CssClass="span-4 last" runat="server"
Text="Disable Inventory:" />
        <asp:checkbox ID="uxInventoryDissabled"  runat="server"
```

```
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {

            Ektron.Cms.Framework.Commerce.InventoryManager Inventorymanager = new
Ektron.Cms.Framework.Commerce.InventoryManager();

            Ektron.Cms.Commerce.InventoryData inventorydata = new
Ektron.Cms.Commerce.InventoryData();
            inventorydata.EntryId = long.Parse(uxEntryId.Text);
            inventorydata.ReorderLevel = int.Parse(uxReorderLevel.Text);
            inventorydata.UnitsInStock = int.Parse(uxUnitsInStock.Text);
            inventorydata.DisableEntryInventoryManagement =
(Boolean)uxInventoryDissabled.Checked;

            inventorydata = Inventorymanager.Add(inventorydata);
            MessageUtilities.UpdateMessage(uxMessage, "Inventory data is added
successfully ", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

DecreaseStockLevel

```
DecreaseStockLevel(System.Int64,System.Int32)
```

To decrease the stock level based on the entry ID and quantity.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry ID
- * Quantity

Parameters

- `entryId`. ID of the catalog entry.
- `quantity`. Quantity of the catalog product.

.aspx code snippet

```
<ol class="formFields">

    <li class="clearfix">
        <ektronUI:Label ID="uxEntryidLabel" AssociatedControlID="uxEntryId"
CssClass="span-3 last" runat="server" Text="* Entry Id:" />
        <ektronUI:TextField  ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQuantityLabel" AssociatedControlID="uxquantity"
CssClass="span-3 last" runat="server" Text="* Quantity:" />
        <ektronUI:TextField  ID="uxquantity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="DecreaseStockLevel"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            Ektron.Cms.Framework.Commerce.InventoryManager Inventorymanager = new
Ektron.Cms.Framework.Commerce.InventoryManager();
            long EntryId = long.Parse(uxEntryId.Text);
            int Quantity = int.Parse(uxquantity.Text);
            Inventorymanager.DecreaseStockLevel(EntryId, Quantity);

            MessageUtilities.UpdateMessage(uxMessage, "Stock level decreased for entry
Id " + EntryId, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the inventory data by its entry ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry ID

Parameters

- `entryId`. ID of the catalog entry.

Returns

InventoryData object.

.aspx code snippet

```
<ol class="formFields">

        <li class="clearfix">
        <ektronUI:Label ID="uxEntryidLabel" AssociatedControlID="uxEntryId"
CssClass="span-3 last" runat="server" Text="* Entry Id:" />
        <ektronUI:TextField  ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="Message1" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUnitsisStock" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUnitsonOrder" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxReorderLevel" runat="server"></asp:Literal>
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.InventoryManager Inventorymanager = new
Ektron.Cms.Framework.Commerce.InventoryManager();
            long EntryId = long.Parse(uxEntryId.Text);
            Ektron.Cms.Commerce.InventoryData inventorydata ;
            inventorydata = Inventorymanager.GetItem(EntryId);

            if (inventorydata != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Inventory Details with Id " +
inventorydata.EntryId, Message.DisplayModes.Success);
                uxId.Text = "EntryId : " + inventorydata.EntryId.ToString();
                uxUnitsisStock.Text = "Unit In Stock : " +
inventorydata.UnitsInStock.ToString();
                uxUnitsonOrder.Text = "Units On Order : " +
inventorydata.UnitsOnOrder.ToString();
```

```
                uxReorderLevel.Text = "Reorder level : " +
inventorydata.ReorderLevel.ToString();

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "No inventory data is exists
to the entry " + EntryId, Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.InventoryCriteria)
```

Retrieves the list of inventorydata based on the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Inventory Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter and sort inventory.

Returns

List of InventoryData objects that matches the criteria.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxInventoryPropertyLabel"
AssociatedControlID="uxInventoryProperty" CssClass="span-4 last" runat="server"
Text="InventoryProperty:" />
```

```
        <asp:DropDownList ID="uxInventoryProperty" runat="server">
            <asp:ListItem>EntryId</asp:ListItem>
            <asp:ListItem>EntryType</asp:ListItem>
            <asp:ListItem>UnitsInStock</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCouponListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       EntryTitle
                    </th>
                    <th>
                        UnitsInStock
                    </th>
                    <th>
                        UnitsOnOrder
                    </th>
                    <th>
                        ReorderLevel
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("EntryTitle")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UnitsInStock")%>
```

```
            </td>
            <td class="devsite-method">
                <%# Eval("UnitsOnOrder")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ReorderLevel")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            InventoryCriteria criteria = new InventoryCriteria();

            if (uxInventoryProperty.SelectedItem.Text == "EntryId")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(InventoryProperty.EntryId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxInventoryProperty.SelectedItem.Text == "UnitsInStock")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(InventoryProperty.UnitsInStock,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(InventoryProperty.UnitsOnOrder,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            Ektron.Cms.Framework.Commerce.InventoryManager inventoryManger = new
Ektron.Cms.Framework.Commerce.InventoryManager();
            List<InventoryData> inventoryList = inventoryManger.GetList(criteria);
```

```
            uxCouponListView.Visible = true;
            uxCouponListView.DataSource = inventoryList;
            uxCouponListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetReorderLevel

```
GetReorderLevel(System.Int64)
```

Get the reorder level based on the entry ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry ID

Parameters

- `entryId`. ID of the catalog entry.

Returns

Returns the reorder level of entry.

.aspx code snippet

```
<ol class="formFields">

        <li class="clearfix">
        <ektronUI:Label ID="uxEntryidLabel" AssociatedControlID="uxEntryId"
CssClass="span-3 last" runat="server" Text="* Entry Id:" />
        <ektronUI:TextField  ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
ReOrderLevel"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
```

```
Required" />
    </li>
</ol>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="Message1" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxReorderLevel" runat="server"></asp:Literal>
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.InventoryManager Inventorymanager = new
Ektron.Cms.Framework.Commerce.InventoryManager();
            long EntryId = long.Parse(uxEntryId.Text);
            long reorderlevel = 0;
            reorderlevel = Inventorymanager.GetReorderLevel(EntryId);

            MessageUtilities.UpdateMessage(uxMessage, "Re-Order Level for the entry Id "
+ EntryId, Message.DisplayModes.Success);
            uxId.Text = "EntryId : " + EntryId.ToString();
            uxReorderLevel.Text = "Re-Order Level : " + reorderlevel.ToString();
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetUnitsInStock

```
GetUnitsInStock(System.Int64)
```

Get the units in stock based on entry ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry ID

Parameters

- `entryId`. ID of the catalog entry.

Returns

Returns the number of units in stock.

.aspx code snippet

```
<ol class="formFields">

        <li class="clearfix">
        <ektronUI:Label ID="uxEntryidLabel" AssociatedControlID="uxEntryId"
CssClass="span-3 last" runat="server" Text="* Entry Id:" />
        <ektronUI:TextField  ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
UnitsInStock"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="Message1" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUnitsInStock" runat="server"></asp:Literal>
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.InventoryManager Inventorymanager = new
Ektron.Cms.Framework.Commerce.InventoryManager();
            long EntryId = long.Parse(uxEntryId.Text);
            long unitsinStock = 0;
            unitsinStock = Inventorymanager.GetUnitsInStock(EntryId);
            MessageUtilities.UpdateMessage(uxMessage, "Units in stock by Entry Id " +
EntryId, Message.DisplayModes.Success);
            uxId.Text = "EntryId : " + EntryId.ToString();
            uxUnitsInStock.Text = "Unit In Stock : " + unitsinStock.ToString();
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetUnitsOnOrder

```
GetUnitsOnOrder(System.Int64)
```

Get the units on order based on entry ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry ID

Parameters

- `entryId`. ID of the catalog entry.

Returns

Returns the number of units on order.

.aspx code snippet

```
<ol class="formFields">

        <li class="clearfix">
        <ektronUI:Label ID="uxEntryidLabel" AssociatedControlID="uxEntryId"
CssClass="span-3 last" runat="server" Text="* Entry Id:" />
        <ektronUI:TextField  ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
UnitsOnOrder"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="Message1" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUnitsonOrder" runat="server"></asp:Literal>
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.InventoryManager Inventorymanager = new
Ektron.Cms.Framework.Commerce.InventoryManager();
            long EntryId = long.Parse(uxEntryId.Text);
            long unitsOnOrder = 0;
            unitsOnOrder = Inventorymanager.GetUnitsOnOrder(EntryId);
            MessageUtilities.UpdateMessage(uxMessage, "Units on Order for the Entry Id "
+ EntryId, Message.DisplayModes.Success);
            uxId.Text = "EntryId : " + EntryId.ToString();
```

```
            uxUnitsonOrder.Text = "Unit On Order : " + unitsOnOrder.ToString();
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IncreaseStockLevel

```
IncreaseStockLevel(System.Int64,System.Int32)
```

To increase the Stock level based on the entryId and quantity.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry ID
- * Quantity

Parameters

- `entryId`. ID of the catalog entry.
- `quantity`. Quantity of the catalog product.

.aspx code snippet

```
<ol class="formFields">

    <li class="clearfix">
        <ektronUI:Label ID="uxEntryidLabel" AssociatedControlID="uxEntryId"
CssClass="span-3 last" runat="server" Text="* Entry Id:" />
        <ektronUI:TextField  ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQuantityLabel" AssociatedControlID="uxquantity"
CssClass="span-3 last" runat="server" Text="* Quantity:" />
        <ektronUI:TextField  ID="uxquantity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
```

```
Text="IncreaseStockLevel"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.InventoryManager Inventorymanager = new
Ektron.Cms.Framework.Commerce.InventoryManager();
            long EntryId = long.Parse(uxEntryId.Text);
            int Quantity = int.Parse(uxquantity.Text);
            Inventorymanager.IncreaseStockLevel(EntryId, Quantity);

            MessageUtilities.UpdateMessage(uxMessage, "Stock level increased for entry
Id " + EntryId, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

IsItemAvailable

```
IsItemAvailable(System.Int64,System.Int32,System.Boolean)
```

Checks whether the number of items are available in the stock.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry ID
- * Quantity
- Disable Inventory

Parameters

- `entryId`. ID of the catalog entry.
- `quantity`. Quantity of the catalog product.
- `checkInventoryDisabled`. True if the inventory is disabled; otherwise false.

Returns

Returns true if the stock matches the quantity; otherwise false.

.aspx code snippet

```
<ol class="formFields">

    <li class="clearfix">
        <ektronUI:Label ID="uxEntryidLabel" AssociatedControlID="uxEntryId"
CssClass="span-3 last" runat="server" Text="* Entry Id:" />
        <ektronUI:TextField  ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQuantityLabel" AssociatedControlID="uxquantity"
CssClass="span-3 last" runat="server" Text="* Entry Id:" />
        <ektronUI:TextField  ID="uxquantity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxInventoryDissabledLable"
AssociatedControlID="uxInventoryDissabled" CssClass="span-3 last" runat="server" Text="*
Disable Inventory:" />
        <asp:checkbox  ID="uxInventoryDissabled"  runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="IsItemAvailable"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            Ektron.Cms.Framework.Commerce.InventoryManager Inventorymanager = new
Ektron.Cms.Framework.Commerce.InventoryManager();
            long EntryId = long.Parse(uxEntryId.Text);
            int Quantity = int.Parse(uxquantity.Text);
                      Boolean Isinventorydissabled = (Boolean)uxInventoryDissabled.Checked;
            Boolean isItemavailable = false;
            isItemavailable = Inventorymanager.IsItemAvailable(EntryId, Quantity,
Isinventorydissabled);
            if (isItemavailable)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Item is available for the
entry Id " + EntryId, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Item not available for the
entry Id " + EntryId, Message.DisplayModes.Information);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.InventoryData)
```

Updates the inventory data to catalog entry item based on its entry ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry ID
- * Units in Stock
- * Reorder Level

Parameters

- `inventoryData`. Inventory data object to be added.

Returns

InventoryData object.

.aspx code snippet

```
<ol class="formFields">

    <li class="clearfix">
        <ektronUI:Label ID="uxEntryidLabel" AssociatedControlID="uxEntryId"
CssClass="span-4 last" runat="server" Text="* Entry Id:" />
        <ektronUI:TextField  ID="uxEntryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQuantityLabel" AssociatedControlID="uxUnitsInStock"
CssClass="span-4 last" runat="server" Text="* Units In Stock:" />
        <ektronUI:TextField  ID="uxUnitsInStock" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxReorderLevelLabel" AssociatedControlID="uxReorderLevel"
CssClass="span-4 last" runat="server" Text="* Reorder Level:" />
        <ektronUI:TextField  ID="uxReorderLevel" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">

        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            Ektron.Cms.Framework.Commerce.InventoryManager inventoryManager = new
Ektron.Cms.Framework.Commerce.InventoryManager();
            long entryId=long.Parse(uxEntryId.Text);
            InventoryData inventoryData = inventoryManager.GetItem(entryId);
            if (inventoryData != null)
            {
                inventoryData.ReorderLevel = int.Parse(uxReorderLevel.Text);
                inventoryData.UnitsInStock = int.Parse(uxUnitsInStock.Text);
```

```
                inventoryData = inventoryManager.Update(inventoryData);
                MessageUtilities.UpdateMessage(uxMessage, "Inventory data updated
successfully ", Message.DisplayModes.Success);
                uxPageMultiView.SetActiveView(uxViewMessage);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Invalid entry Id has been
passed", Message.DisplayModes.Error);
                uxPageMultiView.SetActiveView(uxViewMessage);
            }
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

InventoryCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `InventoryCriteria()`

  ```
  public InventoryCriteria()
  ```

- `InventoryCriteria(Ektron.Cms.Commerce.InventoryProperty, EkEnumeration.OrderByDirection)`

  ```
  public InventoryCriteria(Ektron.Cms.Commerce.InventoryProperty orderByField,
  EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `InventoryProperty` are:

  - EntryId
  - EntryTitle
  - EntryType
  - ReorderLevel
  - UnitsInStock
  - UnitsOnOrder

InventoryData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `DisableEntryInventoryManagement`. Gets or sets the entry disable inventory management flag.

  ```
  public bool DisableEntryInventoryManagement { set; get; }
  ```

- `EntryId`. Gets or sets the ID of the catalog entry (product) that this inventory data represents.

  ```
  public long EntryId { set; get; }
  ```

- `EntryTitle`. Gets or sets the Title of the catalog entry(product) that this inventory data represents.

  ```
  public string EntryTitle { set; get; }
  ```

- `EntryType`. Gets or sets the Type of the catalog entry(product) that this inventory data represents.

  ```
  public EkEnumeration.CatalogEntryType EntryType { set; get; }
  ```

- `InventoryData()`

  ```
  public InventoryData()
  ```

- `InventoryData(long, int, int, int)`

  ```
  public InventoryData(long entryId, int unitsInStock, int unitsOnOrder, int
  reOrderlevel)
  ```

- `ReorderLevel`. Gets or sets the inventory level at which stock should be reordered.

  ```
  public int ReorderLevel { set; get; }
  ```

- `UnitsInStock`. Gets or sets the the number of units currently in stock for the product.

  ```
  public int UnitsInStock { set; get; }
  ```

- `UnitsOnOrder`. Gets or sets The number of units currently on order for this product. This is the number of units that the store has ordered to replenish inventory and NOT the number of units that are currently on order by customers.

  ```
  public int UnitsOnOrder { set; get; }
  ```

OrderManager

9.00 and higher

The class manages commerce orders in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.OrderManager
```

Constructors

- `OrderManager()`
- `OrderManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *SetTrackingNumber* on page 618
- *Update* on page 620

ApplyCoupon

```
ApplyCoupon(System.Int64,System.Int64)
```

Applies a coupon to a specific order.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Order ID
- * Coupon ID

Parameters

- `orderId`. ID of the order.
- `couponId`. ID of the coupon.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-3 last" runat="server" Text="* Order Id:" />
        <ektronUI:TextField  ID="uxOrderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCouponIdLabel" AssociatedControlID="uxCouponId"
CssClass="span-3 last" runat="server" Text="* Coupon Id:" />
        <ektronUI:TextField  ID="uxCouponId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="ApplyCoupon"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            long orderId = long.Parse(uxOrderId.Text);
            long CouponId = long.Parse(uxCouponId.Text);
            OrderData orderData = orderManager.GetItem(orderId);
            if (orderData != null && orderData.Id > 0)
            {
                orderManager.ApplyCoupon(orderId, CouponId);
                MessageUtilities.UpdateMessage(uxMessage, "Coupon with Id " + CouponId +
" is applied to Order "+orderId, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
does not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Capture (order)

```
Capture(System.Int64)
```

Captures the payment for an order.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Order ID

Parameters

- `orderId`. ID of the order.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-3 last" runat="server" Text="* Order Id:" />
        <ektronUI:TextField  ID="uxOrderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Capture"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            long orderId = long.Parse(uxOrderId.Text);
            OrderData orderData = orderManager.GetItem(orderId);
            if (orderData != null && orderData.Id > 0)
            {
```

```
                orderManager.Capture(orderId);
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
is captured.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
does not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Capture (payment)

```
Capture(System.Int64,System.Int64)
```

Captures the payment for an order.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Order ID
- * Payment ID

Parameters

- `orderId`. ID of an order.
- `paymentId`. ID of the payment.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-3 last" runat="server" Text="* Order Id:" />
        <ektronUI:TextField  ID="uxOrderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPaymentIdLabel" AssociatedControlID="uxPaymentId"
CssClass="span-3 last" runat="server" Text="* Payment Id:" />
```

```
        <ektronUI:TextField  ID="uxPaymentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Capture"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            long orderId = long.Parse(uxOrderId.Text);
            long paymentId = long.Parse(uxPaymentId.Text);
            OrderData orderData = orderManager.GetItem(orderId);
            if (orderData != null && orderData.Id > 0)
            {
                orderManager.Capture(orderId,paymentId);
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
is captured.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
does not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes the order by giving the option only to e-commerce administrator or site administrator.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Order ID

Parameters

- `orderId`. ID of the order.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-3 last" runat="server" Text="* Order Id:" />
        <ektronUI:TextField  ID="uxOrderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
```

```
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            long orderId = long.Parse(uxOrderId.Text);
            orderManager.Delete(orderId);
            OrderData orderData = orderManager.GetItem(orderId);
            if (orderData == null || orderData.Id == 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
is deleted.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
is not deleted.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a single order.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Order ID

Parameters

- `orderId`. ID of order to retrieve.

Returns

OrderData object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-3 last" runat="server" Text="* Order Id:" />
        <ektronUI:TextField  ID="uxOrderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxBillingAddressId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxStatus" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCustomerName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCustomerType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCurrenyId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCurrencyName" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            long orderId = long.Parse(uxOrderId.Text);
            OrderData orderData = orderManager.GetItem(orderId);
            if (orderData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for Order with ID " +
orderData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxBillingAddressId.Text = "Billing AddressId : " +
orderData.BillingAddressId;
                uxStatus.Text  = "Status : " + orderData.Status;
                uxCurrencyName.Text = "Customer Name : " + orderData.Customer.UserName;
                uxCustomerType.Text = "Customer Type : " + orderData.CustomerType;
                uxCurrenyId.Text = "Currency Id : " + orderData.Currency.Id;
                uxCurrencyName.Text = "Currency Name : " + orderData.Currency.Name;

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
does not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetList

```
GetList(Ektron.Cms.Commerce.OrderCriteria)
```

Retrieves a list of orders.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Order Property
- * Object Value

Parameters

- `criteria`. Criteria by which to retrieve orders.

Returns

List of OrderData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxOrderPropertyLabel" AssociatedControlID="uxOrderProperty"
CssClass="span-4 last" runat="server" Text="Order Property:" />
        <asp:DropDownList ID="uxOrderProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
            <asp:ListItem>AddressId </asp:ListItem>
            <asp:ListItem>Currency </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxOrderListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Id
                    </th>
                    <th>
                       BillingId
                    </th>
                    <th>
```

```
                        Customer Name
                    </th>
                    <th>
                        Currency Name
                    </th>
                    <th>
                         Order Total
                    </th>
                    <th>
                        CouponUsed
                    </th>
                    <th>
                        WorkflowId
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("BillingAddressId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Customer.UserName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Currency.Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("OrderTotal")%>
            </td>
            <td class="devsite-method">
                <%# Eval("CouponUsed")%>
            </td>
            <td class="devsite-method">
                <%# Eval("WorkflowId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            OrderCriteria criteria = new OrderCriteria();

            if (uxOrderProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(OrderProperty.Id,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxOrderProperty.SelectedItem.Text == "BillingId")
            {
                Objectvalue =long.Parse(uxObjectValue.Text);
                criteria.AddFilter(OrderProperty.BillingId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxOrderProperty.SelectedItem.Text == "CustomerId")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(OrderProperty.CustomerId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue =int.Parse(uxObjectValue.Text);
                criteria.AddFilter(OrderProperty.CurrencyId  ,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            List<OrderData> orderList = orderManager.GetList(criteria);

            uxOrderListView.Visible = true;
            uxOrderListView.DataSource = orderList;
            uxOrderListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

GetOrderPaymentList

```
GetOrderPaymentList(Ektron.Cms.Commerce.OrderPaymentCriteria)
```

Returns the list of order payment data.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Order Payment Property
- * Object Value

Parameters

- `criteria`. Criteria by which to retrieve orders.

Returns

List of Order Payment Data.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxOrderPaymentPropertyLabel"
AssociatedControlID="uxOrderPaymentProperty" CssClass="span-4 last" runat="server"
Text="OrderPayment Property:" />
        <asp:DropDownList ID="uxOrderPaymentProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>OrderId</asp:ListItem>
            <asp:ListItem>PaymentGateway </asp:ListItem>
            <asp:ListItem>CurrencyId </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetOrderPaymentList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
```

```
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxOrderListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Id
                    </th>
                    <th>
                       OrderId
                    </th>
                    <th>
                       Gateway
                    </th>
                    <th>
                       Last4Digits
                    </th>
                    <th>
                       PaymentType
                    </th>
                    <th>
                       PaymentDate
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("OrderId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Gateway")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Last4Digits")%>
            </td>
            <td class="devsite-method">
                <%# Eval("PaymentType")%>
            </td>
```

```
            <td class="devsite-method">
                <%# Eval("PaymentDate")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            OrderPaymentCriteria criteria = new OrderPaymentCriteria();

            if (uxOrderPaymentProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(OrderPaymentProperty.Id,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxOrderPaymentProperty.SelectedItem.Text == "OrderId")
            {
                Objectvalue =long.Parse(uxObjectValue.Text);
                criteria.AddFilter(OrderPaymentProperty.OrderId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxOrderPaymentProperty.SelectedItem.Text == "PaymentGateway")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(OrderPaymentProperty.PaymentGateway,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue =int.Parse(uxObjectValue.Text);
                criteria.AddFilter(OrderPaymentProperty.CurrencyId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
```

```
            List<OrderPaymentData> orderPaymentList = orderManager.GetOrderPaymentList
(criteria);

            uxOrderPaymentListView.Visible = true;
            uxOrderPaymentListView.DataSource = orderPaymentList;
            uxOrderPaymentListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetStatus

```
GetStatus(System.Int64)
```

Returns the status of an order.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Order ID

Parameters

- `orderId`. ID of order to retrieve status for.

Returns

Order status.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-3 last" runat="server" Text="* Order Id:" />
        <ektronUI:TextField  ID="uxOrderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetStatus"></ektronUI:Button>
```

```
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            long orderId = long.Parse(uxOrderId.Text);
            OrderData orderData = orderManager.GetItem(orderId);
            if (orderData != null)
            {
                Ektron.Cms.Common.EkEnumeration.OrderStatus status =
orderManager.GetStatus(orderId);
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
status is "+status +".", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
does not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

PlaceOrder

```
PlaceOrder
```

```
(System.Int64,System.Int64,System.Int64,System.Int64,System.Int64,Ektron.Cms.Commerce.PaymentMethod,System.String,System.String)
```

Create an order to the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Basket ID
- * Customer ID
- * Billing Address ID
- * Shipping Address ID
- * Shipping Method ID
- Special Instruction
- HTTP Host

Payment Method

- * Billing First Name
- * Billing Last Name
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID

Parameters

- `basketId`. ID of the basket.
- `customerId`. ID of the customer.
- `billingAddressId`. ID of billing address.
- `shippingAddressId`. ID of the shipping address.
- `shippingMethodId`. ID of the shipping method.
- `orderPayment`. PaymentMethod object.
- `specialInstructions`. Special instructions.
- `httpHost`. Host URL.

Returns

Placed OrderData.

.aspx code snippet

```
<span>Dimensions</span>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxBasketIdLabel" AssociatedControlID="uxBasketId"
CssClass="span-3 last"
            runat="server" Text="* Basket Id :" />
        <ektronUI:TextField ID="uxBasketId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCustomerIdLabel" AssociatedControlID="uxCustomerId"
CssClass="span-3 last"
            runat="server" Text="* Customer Id :" />
        <ektronUI:TextField ID="uxCustomerId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxBillingAddressIdLagel"
AssociatedControlID="uxBillingAddressId" CssClass="span-3 last"
            runat="server" Text="* Billing AddressId:" />
        <ektronUI:TextField ID="uxBillingAddressId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingAddressIdLabel"
AssociatedControlID="uxShippingAddressId" CssClass="span-3 last"
            runat="server" Text="* Shipping AddressId:" />
        <ektronUI:TextField ID="uxShippingAddressId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingMethodIdLabel"
AssociatedControlID="uxShippingMethodId" CssClass="span-3 last"
            runat="server" Text="* Shipping MethodId :" />
        <ektronUI:TextField ID="uxShippingMethodId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
            <ektronUI:Label ID="uxSpecialInstructionLabel"
AssociatedControlID="uxSpecialInstruction" CssClass="span-3 last" runat="server" Text="
SpecialInstruction:" />
            <ektronUI:TextField ID="uxSpecialInstruction" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
        </li>

        <li class="clearfix">
            <ektronUI:Label ID="uxHttpHostLabel" AssociatedControlID="uxHttpHost"
CssClass="span-3 last" runat="server" Text=" Http Host :" />
            <ektronUI:TextField ID="uxHttpHost" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
        </li>

    <li class="clearfix"><b>Payment Method</b></li>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxBillingFirstNameLabel"
AssociatedControlID="uxBillingFirstName" CssClass="span-3 last"
            runat="server" Text="* Billing FirstName :" />
        <ektronUI:TextField ID="uxBillingFirstName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Label ID="uxBillingLastNameLabel"
AssociatedControlID="uxBillingLastName" CssClass="span-3 last" runat="server" Text="*
Billing LastName:" />
        <ektronUI:TextField  ID="uxBillingLastName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressLine1Label" AssociatedControlID="uxAddressLine1"
CssClass="span-3 last"
            runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxAddressLine1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCityLabel" AssociatedControlID="uxCity" CssClass="span-3
last"
            runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLagel" AssociatedControlID="uxRegionId"
CssClass="span-3 last"
            runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPostalCodeLabel" AssociatedControlID="uxPostalCode"
CssClass="span-3 last"
            runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxPostalCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIdLabel" AssociatedControlID="uxCountryId"
CssClass="span-3 last"
            runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
            <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
            <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server"
Text="* - Required" />
        </li>
</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.OrderManager OrderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            RegionManager regionManager = new RegionManager();
            CountryManager countryManager = new CountryManager();
            OrderData OrderData = null;

            long regionId = long.Parse(uxRegionId.Text);
            int countryId = int.Parse(uxCountryId.Text);
            long basketId=long.Parse(uxBasketId.Text);
            long customerId=long.Parse(uxCustomerId.Text);
            long billingAddressId=long.Parse(uxBillingAddressId.Text);
            long shippingAddressId=long.Parse(uxShippingAddressId.Text);
            long shippingMethodId=long.Parse(uxShippingMethodId.Text);

            RegionData regionData = regionManager.GetItem(regionId);
            CountryData countryData = countryManager.GetItem(countryId);
            AddressData addressData = new AddressData()
            {

                AddressLine1 = uxAddressLine1.Text,
                City = uxCity.Text,
                PostalCode = uxPostalCode.Text,
                Region = regionData,
                Country = countryData,
            };
            PaymentMethod paymentMethod = new PaymentMethod()
            {
                BillingAddress = addressData,
                BillingFirstName = uxBillingFirstName.Text,
                BillingLastName = uxBillingLastName.Text
            };
            OrderData returnOrderData = OrderManager.PlaceOrder(basketId, customerId,
billingAddressId, shippingAddressId, shippingMethodId, paymentMethod,
uxSpecialInstruction.Text, uxHttpHost.Text);
```

```
            if(returnOrderData.Id>0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order is placed with Id " +
OrderData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order is not placed",
Message.DisplayModes.Warning );
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

SetFraud

```
SetFraud(System.Int64)
```

Marks an order as fraudulent.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Order ID

Parameters

- `orderId`. ID of the order.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-3 last" runat="server" Text="* Order Id:" />
        <ektronUI:TextField  ID="uxOrderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="SetFraud"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
```

```
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            long orderId = long.Parse(uxOrderId.Text);
            OrderData orderData = orderManager.GetItem(orderId);
            if (orderData != null && orderData.Id > 0)
            {
                orderManager.SetFraud(orderId);
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
is set as fraud.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
is not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

SetHold

```
SetHold(System.Int64)
```

Sets an order to OnHold.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Order ID

Parameters

- `orderId`. ID of the order.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-3 last" runat="server" Text="* Order Id:" />
        <ektronUI:TextField  ID="uxOrderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="SetHold"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
```

```
Ektron.Cms.Framework.Commerce.OrderManager();
            long orderId = long.Parse(uxOrderId.Text);
            OrderData orderData = orderManager.GetItem(orderId);
            if (orderData != null && orderData.Id > 0)
            {
                orderManager.SetHold(orderId);
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
is set to hold.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
is not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

SetStatus

```
SetStatus(System.Int64,Ektron.Cms.Common.EkEnumeration.OrderStatus)
```

Sets the status of an order.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Order ID
- Order Status

Parameters

- `orderId`. ID of the order.
- `status`. Change the order to the specified status.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
       <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-3 last" runat="server" Text="* Order Id:" />
        <ektronUI:TextField  ID="uxOrderId" CssClass="span-6" runat="server"
```

```
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxOrderStatusLabel" AssociatedControlID="uxOrderStatus"
CssClass="span-4 last" runat="server" Text="Order Status:" />
        <asp:DropDownList ID="uxOrderStatus" runat="server">
            <asp:ListItem value="0">Received</asp:ListItem>
            <asp:ListItem value="1">InProcess</asp:ListItem>
            <asp:ListItem value="2">Shipped </asp:ListItem>
            <asp:ListItem value="3">Completed </asp:ListItem>
            <asp:ListItem value="4">Cancelled </asp:ListItem>
            <asp:ListItem value="5">OnHold </asp:ListItem>
            <asp:ListItem value="6">Fraud </asp:ListItem>
        </asp:DropDownList>
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="SetStatus"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            long orderId = long.Parse(uxOrderId.Text);
            Ektron.Cms.Common.EkEnumeration.OrderStatus orderStatus =
(Ektron.Cms.Common.EkEnumeration.OrderStatus)int.Parse(uxOrderStatus.SelectedValue);
            OrderData orderData = orderManager.GetItem(orderId);
            if (orderData != null && orderData.Id > 0)
            {
                orderManager.SetStatus(orderId,orderStatus);
                MessageUtilities.UpdateMessage(uxMessage, "Order " + orderId + " status
has been set to "+ orderStatus +".", Message.DisplayModes.Success);
            }
```

```
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
is not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

SetTrackingNumber

```
SetTrackingNumber(System.Int64,System.String,System.Boolean)
```

Sets the tracking number for an order part.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Order ID
- * Tracking Number
- Mark As Shipped

Parameters

- `orderId`. ID of order to set tracking number.
- `trackingNumber`. The tracking number
- `markAsShipped`. Mark the item as shipped.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-3 last" runat="server" Text="* Order Id:" />
        <ektronUI:TextField  ID="uxOrderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxOrderTrackingNumberLabel"
AssociatedControlID="uxOrderTrackingNumber" CssClass="span-3 last" runat="server"
Text="* Tracking Number:" />
```

```
        <ektronUI:TextField  ID="uxOrderTrackingNumber" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxMarkedasShippedLabel"
AssociatedControlID="uxMarkedasShipped" CssClass="span-3 last" runat="server" Text="
MarkAsShipped:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxMarkedasShipped"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="SetTrackingNumber"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            bool IsMarked = false;
            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            long orderId = long.Parse(uxOrderId.Text);
            if (uxMarkedasShipped.Checked)
            {
                IsMarked = true;
            }
            OrderData orderData = orderManager.GetItem(orderId);
            if(orderData != null )
            {
                 orderManager.SetTrackingNumber(orderId, uxOrderTrackingNumber.Text,
IsMarked);
                 MessageUtilities.UpdateMessage(uxMessage, "Tracking Number is set to
the order " + orderId , Message.DisplayModes.Success);
            }
            else
```

```
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
does not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.OrderData)
```

Update the existing order.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Order ID
- * Billing ID
- * Customer ID
- * Currency ID
- Order Status
- Client IP
- HTTP Host

Parameters

- `orderData`. OrderData object.

.aspx code snippet

```
<span>Dimensions</span>
<ol class="formFields">

    <li class="clearfix">
        <ektronUI:Label ID="uxOrderIdLabel" AssociatedControlID="uxOrderId"
CssClass="span-3 last"
            runat="server" Text="* Order Id :" />
        <ektronUI:TextField ID="uxOrderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxBillingIdLabel" AssociatedControlID="uxBillingId"
CssClass="span-3 last"
            runat="server" Text="* Billing Id :" />
        <ektronUI:TextField ID="uxBillingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCustomerIdLabel" AssociatedControlID="uxCustomerId"
CssClass="span-3 last"
            runat="server" Text="* Customer Id :" />
        <ektronUI:TextField ID="uxCustomerId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCurrencyIdLagel" AssociatedControlID="uxCurrencyId"
CssClass="span-3 last"
            runat="server" Text="* Currency Id :" />
        <ektronUI:TextField ID="uxCurrencyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxOrderStatusLabel" AssociatedControlID="uxOrderStatus"
CssClass="span-3 last" runat="server" Text="Order Status:" />
        <asp:DropDownList ID="uxOrderStatus" runat="server">
            <asp:ListItem Value ="0">Received</asp:ListItem>
            <asp:ListItem Value ="1">InProcess</asp:ListItem>
            <asp:ListItem Value ="2">Shipped</asp:ListItem>
            <asp:ListItem Value ="3">Completed</asp:ListItem>
            <asp:ListItem Value ="4">Cancelled</asp:ListItem>
            <asp:ListItem Value ="5">OnHold</asp:ListItem>
            <asp:ListItem Value ="6">Fraud</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxClientIpLabel" AssociatedControlID="uxClientIp"
CssClass="span-3 last"
            runat="server" Text=" Client IP :" />
        <ektronUI:TextField ID="uxClientIp" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxHttpHostLabel" AssociatedControlID="uxHttpHost"
CssClass="span-3 last"
            runat="server" Text=" HttpHost :" />
        <ektronUI:TextField ID="uxHttpHost" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long orderId = long.Parse(uxOrderId.Text);
            int currencyId = int.Parse(uxCurrencyId.Text);
            long customerId = long.Parse(uxCustomerId.Text);
            long billingId = long.Parse(uxBillingId.Text);
            Ektron.Cms.Common.EkEnumeration.OrderStatus status =
(Ektron.Cms.Common.EkEnumeration.OrderStatus)int.Parse(uxOrderStatus.Text);

            Ektron.Cms.Framework.Commerce.OrderManager orderManager = new
Ektron.Cms.Framework.Commerce.OrderManager();
            OrderData orderData = orderManager.GetItem(orderId);
            if (orderData != null)
            {
                orderData.BillingAddressId = billingId;
                orderData.Customer.Id = customerId;
                orderData.Currency.Id = currencyId;
                orderData.Status = status;
                orderData.ClientIP = uxClientIp.Text;
                orderData.HttpHost = uxHttpHost.Text;
                orderData=orderManager.Update(orderData);

                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
has been updated.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Order with Id " + orderId + "
does not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

OrderCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `OrderCriteria()`

  ```
  public OrderCriteria()
  ```

- `OrderCriteria(Ektron.Cms.Commerce.OrderProperty, EkEnumeration.OrderByDirection)`

  ```
  public OrderCriteria(Ektron.Cms.Commerce.OrderProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `OrderProperty` are:

  - BillingId
  - ClientIP
  - CouponUsed
  - CustomerBillingID
  - CustomerDateCreated
  - CustomerDisplayName
  - CustomerEmailAddress
  - CustomerEmailAddress2
  - CustomerEmailAddress3
  - CustomerFirstName
  - CustomerId
  - CustomerIsDeleted
  - CustomerLastName
  - CustomerMembershipuser
  - CustomerOrderValue
  - CustomerShippingId
  - CustomerTotalOrders
  - CustomerType
  - CustomerUserName
  - CustomerUserPassword
  - DateCompleted

- DateCreated
- DateRequired
- HasTangibleItems
- Id
- ProductId
- Site
- SpecialInstructions
- StageId
- Status
- Subtotal
- Total
- TotalCoupons
- TotalHandling
- TotalShipping
- TotalShippingTaxes
- TotalTaxes
- VisitorId
- WorkflowId

OrderData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `BillingAddressId`

  ```
  public long BillingAddressId { set; get; }
  ```

- `ClientIP`

  ```
  public string ClientIP { set; get; }
  ```

- `Coupons`

  ```
  public System.Collections.Generic.List<CouponData> Coupons { set; get; }
  ```

- `CouponTotal`

  ```
  public decimal CouponTotal { set; get; }
  ```

- `CouponUsed`

  ```
  public bool CouponUsed { set; get; }
  ```

- `Currency`

  ```
  public Ektron.Cms.Commerce.CurrencyData Currency { set; get; }
  ```

- `Customer`

  ```
  public Ektron.Cms.Commerce.CustomerData Customer { set; get; }
  ```

- CustomerType

```
public EkEnumeration.CustomerType CustomerType { set; get; }
```

- DateCompleted

```
public System.DateTime DateCompleted { set; get; }
```

- DateCreated

```
public System.DateTime DateCreated { set; get; }
```

- DateRequired

```
public System.DateTime DateRequired { set; get; }
```

- HandlingTotal

```
public decimal HandlingTotal { set; get; }
```

- HasTangibleItems

```
public bool HasTangibleItems { set; get; }
```

- HttpHost

```
public string HttpHost { set; get; }
```

- Id

```
public long Id { set; get; }
```

- OrderTotal. Gets or sets the Order Total.

```
public decimal OrderTotal { set; get; }
```

- Parts

```
public System.Collections.Generic.List<OrderPartData> Parts { get; }
```

- Payments

```
public System.Collections.Generic.List<OrderPaymentData> Payments { set; get; }
```

- ShippingTaxTotal. Gets or sets the total shipping tax charged for order.

```
public decimal ShippingTaxTotal { set; get; }
```

- ShippingTotal. Gets or sets the total shipping charged for order.

```
public decimal ShippingTotal { set; get; }
```

- SpecialInstructions

```
public string SpecialInstructions { set; get; }
```

- StageId

```
public int StageId { set; get; }
```

- StageName

```
public string StageName { set; get; }
```

- Status

```
public EkEnumeration.OrderStatus Status { set; get; }
```

- Subtotal. Gets or sets the Order SubTotal.

```
public decimal Subtotal { set; get; }
```

- TaxTotal. Gets or sets the total tax amount for the order.

```
public decimal TaxTotal { set; get; }
```

- `VisitorId`

```
public string VisitorId { set; get; }
```

- `WorkflowId`

```
public System.Guid WorkflowId { set; get; }
```

OrderPaymentCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `OrderPaymentCriteria()`

```
public OrderPaymentCriteria()
```

- `OrderPaymentCriteria(Ektron.Cms.Commerce.OrderPaymentProperty, EkEnumeration.OrderByDirection)`

```
public OrderPaymentCriteria(Ektron.Cms.Commerce.OrderPaymentProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `OrderPaymentProperty` are:

- AuthReceived
- CapturedDate
- CurrencyId
- Id
- Last4Digits
- OrderId
- PaymentDate
- PaymentGateway
- PaymentTotal
- PaymentType
- RecurringTransactionId
- SettledDate
- TransactionId
- VoidedDate

PaymentMethod

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `BillingAddress`. Gets or sets the billing address of the party being authorized.

```
public virtual Ektron.Cms.Commerce.AddressData BillingAddress { set; get; }
```

- `BillingFirstName`. Gets or sets the first name of the billable party.

```
public virtual string BillingFirstName { set; get; }
```

- `BillingLastName`. Gets or sets the last name of the billable party.

```
public virtual string BillingLastName { set; get; }
```

- `GetPayment(System.Collections.Specialized.NameValueCollection)`. Returns a PaymentMethod based upon NameValueCollection. This would be used by the checkout control to get the PaymentMethod posted back from the page.

```
public virtual Ektron.Cms.Commerce.PaymentMethod
  GetPayment(System.Collections.Specialized.NameValueCollection postBackData)
```

- `PaymentMethod()`

```
public PaymentMethod()
```

PackageManager

9.00 and higher

The class manages commerce packages in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.PackageManager
```

Constructors

- `PackageManager()`
- `PackageManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the currently logged-in user identifier.

Methods

- *Add* below
- *Delete* on page 631
- *GetItem* on page 633
- *GetList* on page 635
- *Update* on page 638

Add

```
Add(Ektron.Cms.Commerce.PackageData)
```

Adds a package to the CMS. The Package.Id property is populated with the Package's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

Dimensions

- Units
- * Height
- * Width
- * Length

Weight

- Units
- * Amount
- * Name

Parameters

- `packageData`. PackageData object to add.

Returns

PackageData object.

.aspx code snippet

```
<span>Dimensions</span>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxDimensionUnitLabel" AssociatedControlID="uxDimensionUnit"
CssClass="span-3 last" runat="server" Text=" Units:" />
        <asp:DropDownList ID="uxDimensionUnit" runat="server">
            <asp:ListItem value="0">Inches </asp:ListItem>
            <asp:ListItem value="1">Centimeters</asp:ListItem>
            </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxDimensionHeightLabel"
AssociatedControlID="uxDimensionHeight" CssClass="span-3 last" runat="server" Text="*
Height:" />
        <ektronUI:TextField  ID="uxDimensionHeight" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDimensionWidthLabel"
AssociatedControlID="uxDimensionWidth" CssClass="span-3 last" runat="server" Text="*
Width:" />
```

```
        <ektronUI:TextField  ID="uxDimensionWidth" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDimensionLengthLabel"
AssociatedControlID="uxDimensionLength" CssClass="span-3 last" runat="server" Text="*
Length:" />
        <ektronUI:TextField  ID="uxDimensionLength" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

</ol>
<span>Weight</span>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWeightUnitLabel" AssociatedControlID="uxWeightUnit"
CssClass="span-3 last" runat="server" Text=" Units:" />
        <asp:DropDownList ID="uxWeightUnit" runat="server">
            <asp:ListItem value="0">Pounds</asp:ListItem>
            <asp:ListItem value="1">Kilograms</asp:ListItem>
            </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxWeightAmountLabel" AssociatedControlID="uxWeightAmount"
CssClass="span-3 last" runat="server" Text="* Amount:" />
        <ektronUI:TextField  ID="uxWeightAmount" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

</ol>
<ol class="formFields">

        <li class="clearfix">
        <ektronUI:Label ID="uxPackageNameLabel" AssociatedControlID="uxPackageName"
CssClass="span-3 last" runat="server" Text="* Name:" />
        <ektronUI:TextField  ID="uxPackageName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            PackageManager packageManager = new PackageManager();
            Dimensions dimensions = new Dimensions()
            {
                Height = float.Parse(uxDimensionHeight.Text),
                Length = float.Parse(uxDimensionLength.Text),
                Width = float.Parse(uxDimensionWidth.Text),
                Units = (LinearUnit)int.Parse(uxDimensionUnit.SelectedValue)
            };
            Weight weight = new Weight()
            {
                Amount = float.Parse(uxWeightAmount.Text),
                Units = (WeightUnit)int.Parse(uxWeightUnit.SelectedValue)
            };
            PackageData packageData = new PackageData()
            {
                Dimensions = dimensions,
                MaxWeight = weight,
                Name = uxPackageName.Text
            };
            packageData = packageManager.Add(packageData);
            if (packageData.Id > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Package is added with Id " +
packageData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "package is not added",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Delete

```
Delete(System.Int64)
```

Deletes a package from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Package ID

Parameters

- `id`. ID of package to delete.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxPackageIdLabel" AssociatedControlID="uxPackageId"
CssClass="span-3 last" runat="server" Text="* Package Id:" />
        <ektronUI:TextField  ID="uxPackageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
```

```
        {
            PackageManager packageManager = new PackageManager();
            long packageId = long.Parse(uxPackageId.Text);
            packageManager.Delete(packageId);
            PackageData packageData = packageManager.GetItem(packageId);
            if (packageData == null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Package with Id " + packageId
+ " is deleted.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Package with Id " + packageId
+ " is not deleted.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a package by package ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Package ID

Parameters

- `id`. ID of package to retrieve.

Returns

PackageData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
       <ektronUI:Label ID="uxPackageIdLabel" AssociatedControlID="uxPackageId"
```

```
CssClass="span-3 last" runat="server" Text="* Package Id:" />
        <ektronUI:TextField  ID="uxPackageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxHeigth" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxWidth" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLength" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDimensionUnit" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxWeightAmount" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxWeightUnit" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            PackageManager packageManager = new PackageManager();
            long packageId = long.Parse(uxPackageId.Text);
            PackageData packageData = packageManager.GetItem(packageId);
            if (packageData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for Package with ID "
+ packageData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxName.Text  = "Name : " + packageData.Name;
                uxHeigth.Text  = "Height : " + packageData.Dimensions.Height;
                uxWidth.Text = "Width : " + packageData.Dimensions.Width;
                uxLength.Text = "Length : " + packageData.Dimensions.Length;
                uxDimensionUnit.Text = "LinearUnit : " + packageData.Dimensions.Units;
                uxWeightAmount.Text = "Weigth : " + packageData.MaxWeight.Amount;
                uxWeightUnit.Text = "WeightUnit : " + packageData.MaxWeight.Units;

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Package with Id " + packageId
+ " does not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.PackageCriteria)
```

Retrieves a list of Packages.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Package Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter package being retrieved.

Returns

List of packages meeting criteria filters.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxPackagePropertyLabel"
AssociatedControlID="uxPackageProperty" CssClass="span-4 last" runat="server"
Text="Package Property:" />
        <asp:DropDownList ID="uxPackageProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
            <asp:ListItem>Width </asp:ListItem>
            <asp:ListItem>Weight </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxPackageListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Name
                    </th>
                    <th>
                       Height
                    </th>
                    <th>
                        Width
                    </th>
                    <th>
                       Length
                    </th>
                    <th>
```

```
                        LinearUnit
                    </th>
                    <th>
                        WeightAmount
                    </th>
                    <th>
                        Weightunit
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Dimensions.Height")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Dimensions.Width")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Dimensions.Length")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Dimensions.Units")%>
            </td>
            <td class="devsite-method">
                <%# Eval("MaxWeight.Amount")%>
            </td>
            <td class="devsite-method">
                <%# Eval("MaxWeight.Units")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            PackageCriteria criteria = new PackageCriteria();

            if (uxPackageProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(PackageProperty.Id,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxPackageProperty.SelectedItem.Text == "Name")
            {
                Objectvalue =uxObjectValue.Text;
                criteria.AddFilter(PackageProperty.Name ,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxPackageProperty.SelectedItem.Text == "Width")
            {
                Objectvalue = float.Parse(uxObjectValue.Text);
                criteria.AddFilter(PackageProperty.Length,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue =float.Parse(uxObjectValue.Text);
                criteria.AddFilter(PackageProperty.MaxWeight,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            PackageManager packageManger = new PackageManager();
            List<PackageData> packageList = packageManger.GetList(criteria);

            uxPackageListView.Visible = true;
            uxPackageListView.DataSource = packageList;
            uxPackageListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Update

```
Update(Ektron.Cms.Commerce.PackageData)
```

Updates an existing package in CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Name

Dimensions

- Units
- * Height
- * Width
- * Length

Weight

- Units
- * Amount

Parameters

- `packageData`. PackageData object to save.

Returns

PackageData object.

.aspx code snippet

```
<ol class="formFields">

     <li class="clearfix">
        <ektronUI:Label ID="uxPackageIdLabel" AssociatedControlID="uxPackageId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField  ID="uxPackageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Label ID="uxPackageNameLabel" AssociatedControlID="uxPackageName"
CssClass="span-3 last" runat="server" Text="* Name:" />
        <ektronUI:TextField  ID="uxPackageName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
</ol>
<span>Dimensions</span>
```

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxDimensionUnitLabel" AssociatedControlID="uxDimensionUnit"
CssClass="span-3 last" runat="server" Text=" Units:" />
        <asp:DropDownList ID="uxDimensionUnit" runat="server">
            <asp:ListItem value="0">Inches </asp:ListItem>
            <asp:ListItem value="1">Centimeters</asp:ListItem>
            </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxDimensionHeightLabel"
AssociatedControlID="uxDimensionHeight" CssClass="span-3 last" runat="server" Text="*
Height:" />
        <ektronUI:TextField  ID="uxDimensionHeight" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDimensionWidthLabel"
AssociatedControlID="uxDimensionWidth" CssClass="span-3 last" runat="server" Text="*
Width:" />
        <ektronUI:TextField  ID="uxDimensionWidth" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDimensionLengthLabel"
AssociatedControlID="uxDimensionLength" CssClass="span-3 last" runat="server" Text="*
Length:" />
        <ektronUI:TextField  ID="uxDimensionLength" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

</ol>
<span>Weight</span>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxWeightUnitLabel" AssociatedControlID="uxWeightUnit"
CssClass="span-3 last" runat="server" Text=" Units:" />
        <asp:DropDownList ID="uxWeightUnit" runat="server">
            <asp:ListItem value="0">Pounds</asp:ListItem>
            <asp:ListItem value="1">Kilograms</asp:ListItem>
            </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxWeightAmountLabel" AssociatedControlID="uxWeightAmount"
CssClass="span-3 last" runat="server" Text="* Amount:" />
        <ektronUI:TextField  ID="uxWeightAmount" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>

</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PackageManager packageManager = new PackageManager();
            long packageId=long.Parse(uxPackageId.Text);
            PackageData packageData = packageManager.GetItem(packageId);
            if (packageData != null)
            {
                Dimensions dimensions = packageData.Dimensions;
                dimensions.Height = float.Parse(uxDimensionHeight.Text);
                dimensions.Width = float.Parse(uxDimensionWidth.Text);
                dimensions.Length = float.Parse(uxDimensionLength.Text);
                dimensions.Units = (LinearUnit)int.Parse(uxDimensionUnit.SelectedValue);

                Weight weigth = packageData.MaxWeight;
                weigth.Amount = float.Parse(uxWeightAmount.Text);
                weigth.Units = (WeightUnit)int.Parse(uxWeightUnit.SelectedValue);

                packageData.Name = uxPackageName.Text;
                packageData.Dimensions = dimensions;
                packageData.MaxWeight = weigth;

                packageData = packageManager.Update(packageData);
                MessageUtilities.UpdateMessage(uxMessage, "Package with Id " +
packageData.Id + " is updated.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Package with Id "+ packageId
+" does not exists.", Message.DisplayModes.Warning );
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Data Classes

PackageCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `PackageCriteria()`

  ```
  public PackageCriteria()
  ```

- `PackageCriteria(Ektron.Cms.Commerce.PackageProperty, EkEnumeration.OrderByDirection)`

  ```
  public PackageCriteria(Ektron.Cms.Commerce.PackageProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `PackageProperty` are:

  - Height
  - Id
  - Length
  - LinearUnit
  - MaxWeight
  - Name
  - WeightUnit
  - Width

PackageData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `Dimensions`. Gets or sets the dimensions of the package.

  ```
  public Ektron.Cms.Commerce.Dimensions Dimensions { set; get; }
  ```

- `Id`. Gets or sets the Id of the package.

  ```
  public long Id { set; get; }
  ```

- `MaxWeight`. Gets or sets the maximum weight capacity of the package.

  ```
  public Ektron.Cms.Commerce.Weight MaxWeight { set; get; }
  ```

- `Name`. Gets or sets the name of the package.

```
public string Name { set; get; }
```

- `PackageData()`

```
public PackageData()
```

- `PackageData(string, Ektron.Cms.Commerce.Dimensions, Ektron.Cms.Commerce.Weight)`

```
public PackageData(string name, Ektron.Cms.Commerce.Dimensions dimensions,
Ektron.Cms.Commerce.Weight maxWeight)
```

- `PackageData(string, Ektron.Cms.Commerce.Dimensions, Ektron.Cms.Commerce.Weight, long)`

```
public PackageData(string name, Ektron.Cms.Commerce.Dimensions dimensions,
Ektron.Cms.Commerce.Weight maxWeight, long id)
```

PasswordHistoryManager

9.00 and higher

The class manages password histories.

Namespace

```
Ektron.Cms.Framework.Commerce.PasswordHistoryManager
```

Constructors

- `PasswordHistoryManager()`
- `PasswordHistoryManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Commerce.PasswordHistoryData)
```

Adds a password history to CMS. The PasswordHistory.Id property is populated with the new password history's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Password Hash
- * Change Date Time

Parameters

- `PasswordHistoryData`. PasswordHistoryData object to add.

Returns

PasswordHistoryData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="* UserId :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordHashLabel" AssociatedControlID="uxPasswordHash"
CssClass="span-4 last"
            runat="server" Text="* PasswordHash :" />
        <ektronUI:TextField ID="uxPasswordHash" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxChangeDateLabel" AssociatedControlID="uxChangeDate"
CssClass="span-4 last"
            runat="server" Text="* Change DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxChangeDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxChangeTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
```

```
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxChangePeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PasswordHistoryManager passwordHistoryManager = new PasswordHistoryManager
();
            long userId;
            long.TryParse(uxUserId.Text, out userId);
            string passwordHash = uxPasswordHash.Text;
            string ch = uxChangeDate.Text + " " + uxChangeTime.SelectedItem.Text +
":00:00 " + uxChangePeriod.SelectedItem.Text;
            DateTime changeDate = DateTime.Parse(ch);
            if (userId > 0 && passwordHash != "")
            {
                PasswordHistoryData passwordHistoryData = new PasswordHistoryData();
                passwordHistoryData.UserId = userId;
                passwordHistoryData.PasswordHash = passwordHash;
                passwordHistoryData.ChangeDate = changeDate;
```

```
passwordHistoryManager.RequestInformation.CommerceSettings.PasswordHistory = 1000;
                passwordHistoryData = passwordHistoryManager.Add(passwordHistoryData);
                MessageUtilities.UpdateMessage(uxMessage, "PasswordHistory Added with
Id: " + passwordHistoryData.Id.ToString(), Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a password history from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Password History ID

Parameters

- `id`. ID of PasswordHistoryData object to delete.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordHistoryIdLabel"
AssociatedControlID="uxPasswordHistoryId" CssClass="span-4 last"
            runat="server" Text="* PasswordHistoryId :" />
        <ektronUI:TextField ID="uxPasswordHistoryId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
```

```
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PasswordHistoryManager passwordHistoryManager = new PasswordHistoryManager
();
            long passwordHistoryId;
            long.TryParse(uxPasswordHistoryId.Text, out passwordHistoryId);
            if (passwordHistoryId > 0)
            {
                PasswordHistoryData passwordHistoryData = passwordHistoryManager.GetItem
(passwordHistoryId);
                if (passwordHistoryData != null)
                {
                    passwordHistoryManager.Delete(passwordHistoryId);
                    MessageUtilities.UpdateMessage(uxMessage, "PasswordHistory with Id "
+ passwordHistoryId.ToString() + " has been Deleted.", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
PasswordHistory Id.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
PasswordHistory Id.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a PasswordHistory by ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Password History ID

Parameters

- `id`. ID of PasswordHistory to retrieve.

Returns

PasswordHistoryData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordHistoryIdLabel"
AssociatedControlID="uxPasswordHistoryId" CssClass="span-4 last"
            runat="server" Text="* PasswordHistoryId :" />
        <ektronUI:TextField ID="uxPasswordHistoryId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
```

```
        <asp:Literal ID="uxPasswordHash" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxChangeDate" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PasswordHistoryManager passwordHistoryManager = new PasswordHistoryManager
();
            long passwordHistoryId;
            long.TryParse(uxPasswordHistoryId.Text, out passwordHistoryId);
            if (passwordHistoryId > 0)
            {
                PasswordHistoryData passwordHistoryData = passwordHistoryManager.GetItem
(passwordHistoryId);
                if (passwordHistoryData != null)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "PasswordHistory Details
for Id " + passwordHistoryId.ToString() + " are below", Message.DisplayModes.Success);

                    uxUserId.Text = "UserId : " + passwordHistoryData.UserId.ToString();
                    uxPasswordHash.Text = "PasswordHash : " +
passwordHistoryData.PasswordHash.ToString();
                    uxChangeDate.Text = "ChangeDate : " +
passwordHistoryData.ChangeDate.ToString();
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
PasswordHistory Id.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
PasswordHistory Id.", Message.DisplayModes.Error);
            }
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.PasswordHistoryCriteria)
```

Retrieves a list of password history data objects based upon the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Password History Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter password history being retrieved.

Returns

List of PasswordHistoryData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordHistoryPropertyLabel"
AssociatedControlID="uxPasswordHistoryProperty" CssClass="span-5 last"
            runat="server" Text="* PasswordHistoryProperty :" />
        <asp:DropDownList ID="uxPasswordHistoryProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>UserId</asp:ListItem>
            <asp:ListItem>PasswordHash</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last"
            runat="server" Text="* ObjectValue :" />
```

```
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:ListView ID="uxPasswordHistorylist" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Id
                    </th>
                    <th>
                        Password Hash
                    </th>
                    <th>
                        Change Date
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("PasswordHash")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ChangeDate")%>
            </td>
        </tr>
    </ItemTemplate>
```

```
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PasswordHistoryManager passwordHistoryManager = new PasswordHistoryManager
();
            object objectValue = uxObjectValue.Text;
            string passwordHistoryProperty =
uxPasswordHistoryProperty.SelectedItem.Text;
            PasswordHistoryCriteria criteria = new PasswordHistoryCriteria
(PasswordHistoryProperty.Id, EkEnumeration.OrderByDirection.Ascending);
            if (passwordHistoryProperty == "Id")
            {
                criteria.AddFilter(PasswordHistoryProperty.Id,
CriteriaFilterOperator.EqualTo, objectValue);
            }
            else if (passwordHistoryProperty == "UserId")
            {
                criteria.AddFilter(PasswordHistoryProperty.UserId,
CriteriaFilterOperator.EqualTo, objectValue);
            }
            else
            {
                criteria.AddFilter(PasswordHistoryProperty.PasswordHash,
CriteriaFilterOperator.EqualTo, objectValue);
            }

            List<PasswordHistoryData> passwordHistoryDataList =
passwordHistoryManager.GetList(criteria);
            uxPasswordHistorylist.Visible = true;
            uxPasswordHistorylist.DataSource = passwordHistoryDataList;
            uxPasswordHistorylist.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetRecentPasswords

```
GetRecentPasswords(System.Int64,System.Int32)
```

Retrieves the last N number of passwords for a user.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Number

Parameters

- `userId`. The user ID for whom to retrieve the recent passwords.
- `number`. The number of recent passwords to get.

Returns

List of PasswordHistoryData objects.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="* UserId :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNumberLabel" AssociatedControlID="uxNumber"
CssClass="span-4 last"
            runat="server" Text="* Number :" />
        <ektronUI:TextField ID="uxNumber" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Recent"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
```

```
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxPasswordHistorylist" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        UserId did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Id
                    </th>
                    <th>
                        Password Hash
                    </th>
                    <th>
                        Change Date
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("PasswordHash")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ChangeDate")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PasswordHistoryManager passwordHistoryManager = new PasswordHistoryManager
();
            long userId;
            long.TryParse(uxUserId.Text, out userId);
            int number;
            int.TryParse(uxNumber.Text, out number);
            if (userId > 0 && number > 0)
            {
                List<PasswordHistoryData> passwordHistoryDataList =
passwordHistoryManager.GetRecentPasswords(userId, number);
                uxPasswordHistorylist.Visible = true;
                uxPasswordHistorylist.DataSource = passwordHistoryDataList;
                uxPasswordHistorylist.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

MatchesRecentPassword

```
MatchesRecentPassword(System.Int64,System.String,System.Int32)
```

Returns whether the password matches the user's last N number of passwords.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Password Hash
- * Number

Parameters

- `userId`. The user ID that matches the recent passwords.
- `passwordHash`. Hashed password to match.
- `number`. The number of recent passwords to look in.

Returns

Returns true if a password matches.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="* UserId :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordHashLabel" AssociatedControlID="uxPasswordHash"
CssClass="span-4 last"
            runat="server" Text="* PasswordHash :" />
        <ektronUI:TextField ID="uxPasswordHash" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNumberLabel" AssociatedControlID="uxNumber"
CssClass="span-4 last"
            runat="server" Text="* Number :" />
        <ektronUI:TextField ID="uxNumber" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Match Recent"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PasswordHistoryManager passwordHistoryManager = new PasswordHistoryManager
();
            long userId;
            long.TryParse(uxUserId.Text, out userId);
            int number;
            int.TryParse(uxNumber.Text, out number);
            string passwordHash = uxPasswordHash.Text;
            if (userId > 0 && number > 0 && passwordHash != "")
            {
                bool match = passwordHistoryManager.MatchesRecentPassword(userId,
passwordHash, number);
                if (match)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "PasswordHash " +
passwordHash + " is matched to UserId " + userId.ToString(),
Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "PasswordHash " +
passwordHash + " is not matched to UserId " + userId.ToString(),
Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Purge

```
Purge(System.Int64,System.Int32)
```

Deletes a password history from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Number

Parameters

- `userId`. ID of user to delete password history.
- `number`. Deletes a password history except a given number of recently added passwords.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="* UserId :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNumberLabel" AssociatedControlID="uxNumber"
CssClass="span-4 last"
            runat="server" Text="* Number :" />
        <ektronUI:TextField ID="uxNumber" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Purge"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PasswordHistoryManager passwordHistoryManager = new PasswordHistoryManager
();
            long userId;
            long.TryParse(uxUserId.Text, out userId);
            int number;
            int.TryParse(uxNumber.Text, out number);
            if (userId > 0)
            {
                passwordHistoryManager.Purge(userId, number);
                MessageUtilities.UpdateMessage(uxMessage, "PasswordHistory has been
deleted for UserId " + userId.ToString() + " except " + number.ToString() + " records
which are added recently.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid UserId.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.PasswordHistoryData)
```

Updates an existing password history in CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Password History ID
- User ID
- Password Hash
- * Change Date Time

Parameters

- `PasswordHistoryData`. PasswordHistoryData object to save.

Returns

PasswordHistoryData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordHistoryIdLabel"
AssociatedControlID="uxPasswordHistoryId" CssClass="span-4 last"
            runat="server" Text="* PasswordHistoryId :" />
        <ektronUI:TextField ID="uxPasswordHistoryId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text=" UserId :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordHashLabel" AssociatedControlID="uxPasswordHash"
CssClass="span-4 last"
            runat="server" Text=" PasswordHash :" />
        <ektronUI:TextField ID="uxPasswordHash" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxChangeDateLabel" AssociatedControlID="uxChangeDate"
CssClass="span-4 last"
            runat="server" Text="* Change DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxChangeDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxChangeTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
```

```
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxChangePeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PasswordHistoryManager passwordHistoryManager = new PasswordHistoryManager
();
            long passwordHistoryId;
            long.TryParse(uxPasswordHistoryId.Text, out passwordHistoryId);
            long userId;
            long.TryParse(uxUserId.Text, out userId);
            string passwordHash = uxPasswordHash.Text;
            string ch = uxChangeDate.Text + " " + uxChangeTime.SelectedItem.Text +
":00:00 " + uxChangePeriod.SelectedItem.Text;
            DateTime changeDate = DateTime.Parse(ch);
            if (passwordHistoryId > 0)
            {
                PasswordHistoryData passwordHistoryData = passwordHistoryManager.GetItem
```

```
(passwordHistoryId);
                if(passwordHistoryData != null)
                {
                    passwordHistoryData.UserId = userId > 0 ? userId :
passwordHistoryData.UserId;
                    passwordHistoryData.PasswordHash = passwordHash != "" ? passwordHash
: passwordHistoryData.PasswordHash;
                    passwordHistoryData.ChangeDate = changeDate;
                    passwordHistoryData = passwordHistoryManager.Update
(passwordHistoryData);
                    MessageUtilities.UpdateMessage(uxMessage, "PasswordHistory Updated
for Id: " + passwordHistoryId.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
PasswordHistory Id.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
PasswordHistory Id.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

PasswordHistoryCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `PasswordHistoryCriteria()`

  ```
  public PasswordHistoryCriteria()
  ```

- `PasswordHistoryCriteria (Ektron.Cms.Commerce.PasswordHistoryProperty, EkEnumeration.OrderByDirection)`

```
public
  PasswordHistoryCriteria(Ektron.Cms.Commerce.PasswordHistoryProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `PasswordHistoryProperty` are:

- ChangeDate
- Id
- PasswordHash
- UserId

PasswordHistoryData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `ChangeDate`

  ```
  public System.DateTime ChangeDate { set; get; }
  ```

- `Id`

  ```
  public long Id { set; get; }
  ```

- `PasswordHash`

  ```
  public string PasswordHash { set; get; }
  ```

- `PasswordHistoryData()`

  ```
  public PasswordHistoryData()
  ```

- `PasswordHistoryData(long, string)`

  ```
  public PasswordHistoryData(long userId, string passwordHash)
  ```

- `UserId`

  ```
  public long UserId { set; get; }
  ```

- `Validate`

  ```
  public ValidationResults Validate()
  ```

PaymentGatewayManager

9.00 and higher

The class manages the commerce payment gateway in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.PaymentGatewayManager
```

Constructors

- `PaymentGatewayManager()`
- `PaymentGatewayManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 668
- *GetDefault* on page 669
- *GetItem* on page 671
- *GetList* on page 673
- *SetDefault* on page 676
- *Update* on page 678

Add

```
Add(Ektron.Cms.Commerce.PaymentGatewayData)
```

Adds a payment gateway to the CMS. The PaymentGateway.Id property is populated with the payment gateway's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Name
- User ID
- Password
- Is Default
- Allow Check Payments

Parameters

- `gatewayData`. PaymentGatewayData object to add.

Returns

PaymentGatewayData object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayNameLabel"
AssociatedControlID="uxPaymentGatewayName" CssClass="span-3 last" runat="server" Text="*
Name:" />
        <ektronUI:TextField  ID="uxPaymentGatewayName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayUserIdLabel"
AssociatedControlID="uxPaymentGatewayUserId" CssClass="span-3 last" runat="server"
Text=" User Id:" />
        <ektronUI:TextField  ID="uxPaymentGatewayUserId" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayPasswordLabel"
AssociatedControlID="uxPaymentGatewayPassword" CssClass="span-3 last" runat="server"
Text=" Password:" />
        <ektronUI:TextField  ID="uxPaymentGatewayPassword" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayDefaultLabel"
AssociatedControlID="uxPaymentGatewayDefault" CssClass="span-3 last" runat="server"
Text=" IsDefault:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxPaymentGatewayDefault"
```

```
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayAllowCheckLabel"
AssociatedControlID="uxPaymentGatewayAllowCheckPayments" CssClass="span-3 last"
runat="server" Text=" AllowCheckPayments:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text=""
ID="uxPaymentGatewayAllowCheckPayments" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            PaymentGatewayManager paymentGatewayManager = new PaymentGatewayManager();
            bool isDefault = false;
            bool allowCheckPayment = false;
            if (uxPaymentGatewayDefault.Checked)
            {
                isDefault = true;
            }
            if (uxPaymentGatewayAllowCheckPayments.Checked)
            {
                allowCheckPayment = true;
            }
            PaymentGatewayData paymentGatewayData = new PaymentGatewayData()
            {
                Name = uxPaymentGatewayName.Text,
                UserId = uxPaymentGatewayUserId.Text,
                Password = uxPaymentGatewayPassword.Text,
                IsDefault = isDefault,
```

```
                AllowsCheckPayments = allowCheckPayment
            };

            paymentGatewayData=paymentGatewayManager.Add(paymentGatewayData);
            if (paymentGatewayData.Id > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "PaymentGateway Added with Id
" + paymentGatewayData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "PaymentGateway adding process
is failed", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Delete the existing payment gateway.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `gatewayId`. ID of the gateway to delete.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayIdLabel"
AssociatedControlID="uxPaymentGatewayId" CssClass="span-3 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField  ID="uxPaymentGatewayId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            PaymentGatewayManager paymentGatewayManager = new PaymentGatewayManager();
            long paymentGatewayId = int.Parse(uxPaymentGatewayId.Text);

            paymentGatewayManager.Delete(paymentGatewayId);
            MessageUtilities.UpdateMessage(uxMessage, "PaymentGateway " +
paymentGatewayId + " is deleted", Message.DisplayModes.Success );

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetDefault

```
GetDefault()
```

Returns default payment gateway.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Returns

Default PaymentGatewayData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetDefault"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxPassword" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsDefault" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAllowCreditCards" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            PaymentGatewayManager paymentGatewayManager = new PaymentGatewayManager();
            PaymentGatewayData paymentGatewayData = paymentGatewayManager.GetDefault();

            if (paymentGatewayData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for default payment
gateway with ID " + paymentGatewayData.Id.ToString() + " are below",
Message.DisplayModes.Success);
                uxUserId.Text = "User Id : " + paymentGatewayData.UserId;
                uxPassword.Text = "Password : " + paymentGatewayData.Password;
                uxName.Text = "Name : " + paymentGatewayData.Name;
                uxIsDefault.Text = "IsDefault : " + paymentGatewayData.IsDefault;
                uxAllowCreditCards.Text = "AllowCreditCardPayments : " +
paymentGatewayData.AllowsCheckPayments;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the payment gateway by gateway ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `gatewayId`. ID of the gateway to retrieved.

Returns

PaymentGatewayData object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayIDLabel"
AssociatedControlID="uxPaymentGatewayId" CssClass="span-3 last" runat="server" Text="*
Id :" />
        <ektronUI:TextField ID="uxPaymentGatewayId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxPassword" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsDefault" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAllowCreditCards" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            PaymentGatewayManager paymentGatewayManager = new PaymentGatewayManager();
            long paymentGatewayId=long.Parse(uxPaymentGatewayId.Text);
            PaymentGatewayData paymentGatewayData = paymentGatewayManager.GetItem
(paymentGatewayId);

            if (paymentGatewayData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for PaymentGateway
with ID " + paymentGatewayData.Id.ToString() + " are below",
Message.DisplayModes.Success);
                uxUserId.Text = "User Id : " + paymentGatewayData.UserId;
                uxPassword.Text = "Password : " + paymentGatewayData.Password;
                uxName.Text = "Name : " + paymentGatewayData.Name;
                uxIsDefault.Text = "IsDefault : " + paymentGatewayData.IsDefault;
                uxAllowCreditCards.Text = "AllowCreditCardPayments : " +
paymentGatewayData.AllowsCheckPayments;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "PaymentGateway " +
paymentGatewayId + " does not exist. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.PaymentGatewayCriteria)
```

Retrieves a list of payment gateways.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Payment Gateway Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter payment gateway being retrieved.

Returns

List of payment gateways meeting criteria filters.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayPropertyLabel"
AssociatedControlID="uxPaymentGatewayProperty" CssClass="span-4 last" runat="server"
Text="PaymentGateway Property:" />
        <asp:DropDownList ID="uxPaymentGatewayProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
            <asp:ListItem>IsDefault </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxPaymentGatewayListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Name
                    </th>
                    <th>
                       User Id
                    </th>
                    <th>
                        IsDefault
                    </th>
                    <th>
```

```
                        AllowsCreditCardPayments
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsDefault")%>
            </td>
            <td class="devsite-method">
                <%# Eval("AllowsCreditCardPayments")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            PaymentGatewayCriteria criteria = new PaymentGatewayCriteria();

            if (uxPaymentGatewayProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(PaymentGatewayProperty.Id,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
```

```
            else if (uxPaymentGatewayProperty.SelectedItem.Text == "Name")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(PaymentGatewayProperty.Name,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = bool.Parse(uxObjectValue.Text);
                criteria.AddFilter(PaymentGatewayProperty.IsDefault,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            PaymentGatewayManager paymentGatewayManger = new PaymentGatewayManager();
            List<PaymentGatewayData> paymentGatewayList = paymentGatewayManger.GetList
(criteria);

            uxPaymentGatewayListView.Visible = true;
            uxPaymentGatewayListView.DataSource = paymentGatewayList;
            uxPaymentGatewayListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

SetDefault

```
SetDefault(System.Int64)
```

Mark the existing gateway as default gateway.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Gateway ID

Parameters

- `gatewayId`. ID of the gateway to become default.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayIdLabel"
AssociatedControlID="uxPaymentGatewayId" CssClass="span-3 last" runat="server" Text="*
GatewayId:" />
        <ektronUI:TextField  ID="uxPaymentGatewayId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="SetDefault"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            PaymentGatewayManager paymentGatewayManager = new PaymentGatewayManager();

            long paymentGatewayId = long.Parse(uxPaymentGatewayId.Text);
            PaymentGatewayData paymentGatewayData=paymentGatewayManager.GetItem
(paymentGatewayId);
            if (paymentGatewayData != null)
            {
                paymentGatewayManager.SetDefault(paymentGatewayId);
                MessageUtilities.UpdateMessage(uxMessage, "PaymentGateway " +
paymentGatewayId + " is set as default payment gateway", Message.DisplayModes.Success );
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "PaymentGateway " +
paymentGatewayId + " is does not exists", Message.DisplayModes.Information);
            }
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.PaymentGatewayData)
```

Update the existing PaymentGatewayData object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Name
- User ID
- Password
- Is Default

Parameters

- `gatewayData`. PaymentGatewayData object to update.

Returns

Payment gateway data object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayIdLabel"
AssociatedControlID="uxPaymentGatewayId" CssClass="span-3 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField  ID="uxPaymentGatewayId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayNameLabel"
AssociatedControlID="uxPaymentGatewayName" CssClass="span-3 last" runat="server" Text="*
```

```
Name:" />
        <ektronUI:TextField  ID="uxPaymentGatewayName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayUserIdLabel"
AssociatedControlID="uxPaymentGatewayUserId" CssClass="span-3 last" runat="server"
Text=" User Id:" />
        <ektronUI:TextField  ID="uxPaymentGatewayUserId" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayPasswordLabel"
AssociatedControlID="uxPaymentGatewayPassword" CssClass="span-3 last" runat="server"
Text=" Password:" />
        <ektronUI:TextField  ID="uxPaymentGatewayPassword" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPaymentGatewayDefaultLabel"
AssociatedControlID="uxPaymentGatewayDefault" CssClass="span-3 last" runat="server"
Text=" IsDefault:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxPaymentGatewayDefault"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            PaymentGatewayManager paymentGatewayManager = new PaymentGatewayManager();
            bool isDefault = false;
```

```
            if(uxPaymentGatewayDefault.Checked)
            {
                isDefault  = true ;
            }
            long paymentGatewayId=long.Parse(uxPaymentGatewayId.Text);
            PaymentGatewayData paymentGatewayData = paymentGatewayManager.GetItem
(paymentGatewayId);
            if (paymentGatewayData != null)
            {
                paymentGatewayData.UserId = uxPaymentGatewayUserId.Text;
                paymentGatewayData.Password = uxPaymentGatewayPassword.Text;
                paymentGatewayData.Name = uxPaymentGatewayName.Text;
                paymentGatewayData.IsDefault = isDefault;
                paymentGatewayData=paymentGatewayManager.Update(paymentGatewayData);

                MessageUtilities.UpdateMessage(uxMessage, "PaymentGateway updated with
Id " + paymentGatewayData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "PaymentGateway details
doesn't exists", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

PaymentGatewayCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `PaymentGatewayCriteria()`

  ```
  public PaymentGatewayCriteria()
  ```

- `PaymentGatewayCriteria (Ektron.Cms.Commerce.PaymentGatewayProperty, EkEnumeration.OrderByDirection)`

  ```
  public PaymentGatewayCriteria(Ektron.Cms.Commerce.PaymentGatewayProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

Accepted fields for the `PaymentGatewayProperty` are:

- AllowsCheckPayments
- Id
- IsDefault
- Name

PaymentGatewayData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `AllowsCheckPayments`. Gets or sets whether the payment gateway accepts check payments. * True = the payment gateway accepts check payments * False = the payment gateway does not accept check payments

  ```
  public bool AllowsCheckPayments { set; get; }
  ```

- `AllowsCreditCardPayments`. Gets or sets whether the payment gateway accepts credit cards. * True = the payment gateway accepts credit cards * False = the payment gateway does not accept credit cards

  ```
  public bool AllowsCreditCardPayments { set; get; }
  ```

- `CustomFieldOne`. Gets or sets the first custom field a payment gateway might require. Each type of payment gateway provider accepts configuration parameters. For example, Authorize.NET requires a username and password while PayFlow requires a username, password, vendor, and partner. So, if you were creating an object for PayFlow, you would enter the required vendor information in this property. If your payment gateway does not require additional information, leave blank.

  ```
  public string CustomFieldOne { set; get; }
  ```

- `CustomFieldTwo`. Gets or sets the second custom field a payment gateway might require. Each type of payment gateway provider accepts configuration parameters. For example, Authorize.NET requires a username and password while PayFlow requires a username, password, vendor, and partner. So, if you were creating an object for PayFlow, you would enter the required partner information in this property. If your payment gateway does not require additional information, leave blank.

  ```
  public string CustomFieldTwo { set; get; }
  ```

- `Id`. Gets or sets the Ektron CMS ID for the payment gateway data object. You cannot set the ID of a newly created data object. The ID of an object is set internally by Ektron CMS during the creation (Add) process. You can set the ID of the object for updating purposes.

  ```
  public long Id { set; get; }
  ```

- `IsCustom`. Gets or sets whether this payment gateway is a custom payment gateway or one that comes pre-installed with Ektron CMS. Set this property to

true when creating a custom payment gateway. * True = is a custom payment gateway * False = is not a custom payment gateway

```
public bool IsCustom { set; get; }
```

- `IsDefault`. Gets or sets whether this payment gateway is the default in Ektron CMS. * True = default payment gateway * False = not the default payment gateway.

```
public bool IsDefault { set; get; }
```

- `Name`. Gets or sets the name associated with the payment gateway. For example, if you were adding information for a payment gateway company called EkPay, enter "EkPay" for this property. When creating a payment gateway, a name value is required.

```
public string Name { set; get; }
```

- `Password`. Gets or sets the password associated with a payment gateway. This is not an Ektron CMS password. This information is supplied by the payment gateway.

```
public string Password { set; get; }
```

- `PaymentGatewayData()`

```
public PaymentGatewayData()
```

- `PaymentGatewayData(string, string, string, string, string, bool, bool, bool)`

```
public PaymentGatewayData(string name, string user,
  string password, string custom1, string custom2,
  bool isDefault, bool allowsCreditCards, bool allowsChecks)
```

- `UserId`. Gets or sets the user ID associated with a payment gateway. This is not an Ektron CMS user ID. This information is supplied by the payment gateway.

```
public string UserId { set; get; }
```

- `Validate()`. Validates the PaymentGatewayData object is valid for saving. Returns a ValidationResult object with any validation errors.

```
public ValidationResults Validate()
```

ProductTypeManager

9.00 and higher

The class manipulates the catalog's product type data in CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.ProductTypeManager
```

Constructors

- `ProductTypeManager()`
- `ProductTypeManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Commerce.ProductTypeData)
```

Adds a product type into the CMS. The ProductTypeData.Id property is populated with the product type's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- Description
- Class

Parameters

- `productType`. ProductTypeData object.

Returns

Returns the added ProductTypeData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-4
last"
            runat="server" Text="* Title :" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last"
            runat="server" Text=" Description :" />
        <ektronUI:TextField ID="uxDescription" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <ektronUI:Label ID="uxProductTypeClassLabel"
AssociatedControlID="uxProductTypeClass" CssClass="span-4"
            runat="server" Text="* ProductType Class :" />
       <asp:DropDownList ID="uxProductTypeClass" runat="server">
            <asp:ListItem Value="0">Product</asp:ListItem>
            <asp:ListItem Value="2">Kit</asp:ListItem>
            <asp:ListItem Value="3">Bundle</asp:ListItem>
            <asp:ListItem Value="4">Subscription</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
```

```
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ProductTypeManager productTypeManager = new ProductTypeManager();
            string title = uxTitle.Text;
            if (title != "")
            {
                ProductTypeData productTypeData = new ProductTypeData();
                productTypeData.Title = title;
                productTypeData.Description = uxDescription.Text;
                productTypeData.DefaultXslt = "0";
                productTypeData.EntryClass =
(Ektron.Cms.Common.EkEnumeration.CatalogEntryType) int.Parse
(uxProductTypeClass.SelectedValue);
                productTypeData = productTypeManager.Add(productTypeData);
                MessageUtilities.UpdateMessage(uxMessage, "ProductType Added with Id " +
productTypeData.Id.ToString(), Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Title",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes product type by it's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Product Type ID

Parameters

- `id`. ID of the product type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxProductTypeIdLabel" AssociatedControlID="uxProductTypeId"
CssClass="span-4 last"
            runat="server" Text="* ProductTypeId :" />
        <ektronUI:TextField ID="uxProductTypeId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0"/>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
```

```
        {
            ProductTypeManager productTypeManager = new ProductTypeManager();
            long productTypeId;
            long.TryParse(uxProductTypeId.Text, out productTypeId);
            if (productTypeId > 0)
            {
                ProductTypeData productTypeData = productTypeManager.GetItem
(productTypeId);
                if (productTypeData != null && productTypeData.Id > 0)
                {
                    productTypeManager.Delete(productTypeId);
                    MessageUtilities.UpdateMessage(uxMessage, "ProductType with Id " +
productTypeId.ToString() + " has been Deleted", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
ProductTypeId", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
ProductTypeId", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64,System.Boolean)
```

Retrieves the single product type by its ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Product Type ID
- * Include Attributes

Parameters

- `productTypeId`. The ID of the product type to get.
- `withAttributes`. If it is true, it will fetch attributes; otherwise not.

Returns

Returns ProductTypeData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxProductTypeIdLabel" AssociatedControlID="uxProductTypeId"
CssClass="span-4 last"
            runat="server" Text="* ProductTypeId :" />
        <ektronUI:TextField ID="uxProductTypeId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0"/>
    </li>
    <li class="clearfix">

        <ektronUI:Label ID="uxIncludeAttributesLabel"
AssociatedControlID="uxIncludeAttributes" CssClass="span-4 last"
            runat="server" Text="* Include Attributes:" />
        <asp:checkbox ID="uxIncludeAttributes" runat="server" Checked="True" />
    </li>
    <li class="clearfix">

        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxProductTypeListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="True">
<EmptyDataTemplate ><li>ProductTypeId did not match any
records.</li></EmptyDataTemplate>
    <LayoutTemplate>
        <asp:PlaceHolder ID="aspItemPlaceholder" runat="server"></asp:PlaceHolder>
    </LayoutTemplate>
    <ItemTemplate>
        <ol class="formFields">
            <li class="clearfix">Title : <%# Eval("Title") %> </li>
            <li class="clearfix">Description : <%# Eval("Description") %> </li>
            <li class="clearfix">Date Created : <%# Eval("DateCreated") %> </li>


            <asp:ListView ID="uxAttributesListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" DataSource='<%# Eval("Attributes")%>'>
                <EmptyDataTemplate ><li class="clearfix">No
Attributes</li></EmptyDataTemplate>
                <LayoutTemplate>
```

```
                        <h6>Attributes</h6>
                        <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </LayoutTemplate>
                <ItemTemplate>
                    <li class="clearfix">Attribute Id : <%# Eval("Id") %> </li>
                    <li class="clearfix">Name : <%# Eval("Name") %> </li>
                    <li class="clearfix">Default Value : <%# Eval("DefaultValue")%>
</li>
                </ItemTemplate>
            </asp:ListView>
        </ol>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ProductTypeManager productTypeManager = new ProductTypeManager();
            long productTypeId;
            long.TryParse(uxProductTypeId.Text, out productTypeId);
            if (productTypeId > 0)
            {
                ProductTypeData productTypeData = productTypeManager.GetItem
(productTypeId, uxIncludeAttributes.Checked);
                if (productTypeData != null && productTypeData.Id > 0)
                {
                                        MessageUtilities.UpdateMessage(uxMessage, "ProductType Detail
productTypeData.Id.ToString() + " are below", Message.DisplayModes.Success);

                    uxProductTypeListView.DataSource = new ProductTypeData[]
{productTypeData};
                    uxProductTypeListView.DataBind();
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
ProductTypeId", Message.DisplayModes.Error);
                }
            }
```

```
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
ProductTypeId", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (folder ID)

```
GetList(System.Int64)
```

Retrieves a list of ProductTypeData object by folder ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID

Parameters

- `folderId`. ID of the Folder that uses product types.

Returns

Returns a list of ProductTypeData objects.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-4 last"
            runat="server" Text="* FolderId :" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0"/>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
```

```
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxProductTypelist" runat="server"
ItemPlaceholderID="aspItemPlaceholder">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
                        Description
                    </th>
                    <th>
                        Date Created
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Description")%>
            </td>
            <td class="devsite-method">
                <%# Eval("DateCreated")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ProductTypeManager productTypeManager = new ProductTypeManager();
            long folderId;
            long.TryParse(uxFolderId.Text, out folderId);
            if (folderId > 0)
            {
                List<ProductTypeData> productTypeDataList = productTypeManager.GetList
(folderId);
                uxProductTypelist.DataSource = productTypeDataList;
                uxProductTypelist.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Folder
Id", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (product type criteria)

```
GetList(Ektron.Cms.Commerce.ProductTypeCriteria)
```

Retrieves a list of ProductTypeData objects based upon the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Product Type Property
- * Object Value

Parameters

- `criteria`. Criteria used to retrieve product types.

Returns

Returns a list of ProductTypeData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxProductTypePropertyLabel"
AssociatedControlID="uxProductTypeProperty" CssClass="span-5 last"
            runat="server" Text="* ProductTypeProperty :" />
        <asp:DropDownList ID="uxProductTypeProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Title</asp:ListItem>
            <asp:ListItem>Description</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last"
            runat="server" Text="* ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:ListView ID="uxProductTypelist" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Title
                    </th>
```

```
                    <th>
                        Description
                    </th>
                    <th>
                        Date Created
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Description")%>
            </td>
            <td class="devsite-method">
                <%# Eval("DateCreated")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ProductTypeManager productTypeManager = new ProductTypeManager();
            object objectValue = uxObjectValue.Text;
            string productTypeProperty = uxProductTypeProperty.SelectedItem.Text;
            ProductTypeCriteria criteria = new ProductTypeCriteria
```

```
(ProductTypeProperty.Title, EkEnumeration.OrderByDirection.Ascending);
            if (productTypeProperty == "Id")
            {
                criteria.AddFilter(ProductTypeProperty.Id,
CriteriaFilterOperator.EqualTo, objectValue);
            }
            else if (productTypeProperty == "Title")
            {
                criteria.AddFilter(ProductTypeProperty.Title,
CriteriaFilterOperator.Contains, objectValue);
            }
            else
            {
                criteria.AddFilter(ProductTypeProperty.Description,
CriteriaFilterOperator.Contains, objectValue);
            }

            List<ProductTypeData> productTypeDataList = productTypeManager.GetList
(criteria);
            uxProductTypelist.Visible = true;
            uxProductTypelist.DataSource = productTypeDataList;
            uxProductTypelist.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.ProductTypeData)
```

Updates an existing product type.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Product Type ID
- Type
- Description

Parameters

- `productType`. ProductTypeData object.

Returns

Returns the updated ProductTypeData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxProductTypeIdLabel" AssociatedControlID="uxProductTypeId"
CssClass="span-4 last"
            runat="server" Text="* ProductTypeId :" />
        <ektronUI:TextField ID="uxProductTypeId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0"/>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-4
last"
            runat="server" Text=" Title :" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last"
            runat="server" Text=" Description :" />
        <ektronUI:TextField ID="uxDescription" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ProductTypeManager productTypeManager = new ProductTypeManager();
            long productTypeId;
            long.TryParse(uxProductTypeId.Text, out productTypeId);
            if (productTypeId > 0)
            {
                ProductTypeData productTypeData = productTypeManager.GetItem
(productTypeId);
                if (productTypeData != null && productTypeData.Id > 0)
                {
                    productTypeData.Title = uxTitle.Text != "" ? uxTitle.Text :
productTypeData.Title;
                    productTypeData.Description = uxDescription.Text != "" ?
uxDescription.Text : productTypeData.Description;
                    productTypeManager.Update(productTypeData);
                    MessageUtilities.UpdateMessage(uxMessage, "ProductType Updated for
Id " + productTypeId.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
ProductTypeId", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
ProductTypeId", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

ProductTypeCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `ProductTypeCriteria()`

```
public ProductTypeCriteria()
```

- `ProductTypeCriteria(Ektron.Cms.Commerce.ProductTypeProperty, EkEnumeration.OrderByDirection)`

```
public ProductTypeCriteria(Ektron.Cms.Commerce.ProductTypeProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `ProductTypeProperty` are:

- ConfigClass
- DateCreated
- DefaultXslt
- Description
- EditorFirstName
- EditorLastName
- EditXslt
- FieldList
- Is
- LastEditDate
- LogicalPath
- PackageDisplayXslt
- PackageXslt
- PhysicalPath
- SaveXslt
- SecondaryConfig
- SubscriptionProvider
- Title
- UserId
- XmlAdvConfig
- XmlNamespace
- XmlSchema
- Xslt1
- Xslt2
- Xslt3
- Xslt4
- Xslt5

ProductTypeData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- Attributes

```
public System.Collections.Generic.List<ProductTypeAttributeData>
  Attributes { set; get; }
```

- DefaultThumbnails

```
public System.Collections.Generic.List<ThumbnailDefaultData>
  DefaultThumbnails { set; get; }
```

- EntryClass

```
public EkEnumeration.CatalogEntryType EntryClass { set; get; }
```

- SubscriptonProvider

```
public string SubscriptionProvider { set; get; }
```

- Title

```
public new string Title { set; get; }
```

RecommendationManager

9.00 and higher

The class manipulates the recommended items to the catalog entry.

Namespace

```
Ektron.Cms.Framework.Commerce.RecommendationManager
```

Constructors

- `RecommendationManager()`
- `RecommendationManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Delete* below
- *DeleteByEntry* on page 702
- *GetItem* on page 704
- *GetList* on page 706
- *UpdateCrossSell* on page 709
- *UpdateUpSell* on page 711

Delete

```
Delete(System.Int64)
```

Deletes a recommendation.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Recommendation ID

Parameters

- `recommendationId`. ID of recommendation to delete.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRecommendationIdLabel"
AssociatedControlID="uxRecommendationId" CssClass="span-4 last"
            runat="server" Text="* RecommendationId :" />
        <ektronUI:TextField ID="uxRecommendationId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            RecommendationManager recommendationManager = new RecommendationManager();
            long recommendationId;
            long.TryParse(uxRecommendationId.Text, out recommendationId);
            if (recommendationId > 0)
            {
                RecommendationItemData recommendationItemData =
recommendationManager.GetItem(recommendationId);
                if (recommendationItemData != null && recommendationItemData.Id > 0)
                {
                    recommendationManager.Delete(recommendationId);
                    MessageUtilities.UpdateMessage(uxMessage, "Recommendation is Deleted
for the Id " + recommendationItemData.Id.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
Recommendation Id", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
Recommendation Id", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeleteByEntry

```
DeleteByEntry(System.Int64)
```

Delete recommendations associated with a catalog entry.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry ID

Parameters

- `entryId`. ID of entry for which to delete recommendations.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryId"
CssClass="span-4 last"
            runat="server" Text="* EntryId :" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RecommendationManager recommendationManager = new RecommendationManager();
            long entryId;
            long.TryParse(uxEntryId.Text, out entryId);
            if (entryId > 0)
            {
                CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
                EntryData entryData = catalogEntryManager.GetItem(entryId);
                if (entryData != null && entryData.Id > 0)
                {
                    recommendationManager.DeleteByEntry(entryId);
                    MessageUtilities.UpdateMessage(uxMessage, "Recommendations are
Deleted for the Entry Id " + entryData.Id.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Entry
Id", Message.DisplayModes.Error);
```

```
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Entry Id",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a recommendation by id.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Recommendation ID

Parameters

- `recommendationId`. ID of recommendation to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRecommendationIdLabel"
AssociatedControlID="uxRecommendationId" CssClass="span-4 last"
            runat="server" Text="* RecommendationId :" />
        <ektronUI:TextField ID="uxRecommendationId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>
<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxEntryId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxGroupId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RecommendationManager recommendationManager = new RecommendationManager();
            long recommendationId;
            long.TryParse(uxRecommendationId.Text, out recommendationId);
            if (recommendationId > 0)
            {
                RecommendationItemData recommendationItemData =
recommendationManager.GetItem(recommendationId);
                if (recommendationItemData != null && recommendationItemData.Id > 0)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Recommendation Details
for the Id " + recommendationItemData.Id.ToString(), Message.DisplayModes.Success);
                    uxEntryId.Text = "Entry Id : " +
recommendationItemData.EntryId.ToString();
                    uxGroupId.Text = "Group Id : " +
recommendationItemData.GroupId.ToString();
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
Recommendation Id", Message.DisplayModes.Error);
                }
            }
            else
            {
```

```
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
Recommendation Id", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(System.Int64,System.Int64,Ektron.Cms.Common.EkEnumeration.RecommendationType)
```

Retrieves a list of RecommendationItemData objects based upon the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Entry ID
- * Entry Language
- * Recommendation Type

Parameters

- `entryId`. Catalog entry ID.
- `entryLanguage`. Catalog entry language.
- `type`. Recommendation type.

Returns

Returns list of RecommendationItemData which are matches the criteria.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdLabel" AssociatedControlID="uxEntryId"
CssClass="span-4 last"
            runat="server" Text="* EntryId :" />
        <ektronUI:TextField ID="uxEntryId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryLanguageLabel" AssociatedControlID="uxEntryLanguage"
CssClass="span-4 last"
            runat="server" Text="* EntryLanguage :" />
        <ektronUI:TextField ID="uxEntryLanguage" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRecommendationTypeLabel"
AssociatedControlID="uxRecommendationType"
            CssClass="span-4 last" runat="server" Text="* RecommendationType :" />
        <asp:DropDownList ID="uxRecommendationType" runat="server">
            <asp:ListItem>CrossSell</asp:ListItem>
            <asp:ListItem>UpSell</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxRecommendationList" runat="server"
ItemPlaceholderID="aspItemPlaceholder">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        EntryId
                    </th>
                    <th>
                        GroupId
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
```

```
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("EntryId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("GroupId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RecommendationManager recommendationManager = new RecommendationManager();
            long entryId;
            long.TryParse(uxEntryId.Text, out entryId);
            long entryLanguage;
            long.TryParse(uxEntryLanguage.Text, out entryLanguage);
            string recommendationType = uxRecommendationType.SelectedItem.Text;
            if (entryId > 0 && entryLanguage > 0)
            {
                CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
                EntryData entryData = catalogEntryManager.GetItem(entryId);
                if (entryData != null && entryData.Id > 0)
                {
                    List<RecommendationItemData> recommendationItemDataList = null;
                    if (recommendationType == "CrossSell")
                    {
                        recommendationItemDataList = recommendationManager.GetList
(entryId, entryLanguage, Ektron.Cms.Common.EkEnumeration.RecommendationType.CrossSell);
                    }
                    else
                    {
                        recommendationItemDataList = recommendationManager.GetList
(entryId, entryLanguage, Ektron.Cms.Common.EkEnumeration.RecommendationType.Upsell);
                    }
                    uxRecommendationList.DataSource = recommendationItemDataList;
                    uxRecommendationList.DataBind();
```

```
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Entry
Id", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

UpdateCrossSell

```
UpdateCrossSell(System.Int64,System.Collections.Generic.IEnumerable)
```

Updates the CrossSell.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * CrossSell Entry ID
- * Entry IDs (Comma Separated)

Parameters

- `entryId`. ID of entry.
- `recommendations`. Enumerable of RecommendationItemData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCrossSellEntryIdLabel"
AssociatedControlID="uxCrossSellEntryId" CssClass="span-4 last"
            runat="server" Text="* CrossSell EntryId :" />
        <ektronUI:TextField ID="uxCrossSellEntryId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
```

```
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdsLabel" AssociatedControlID="uxEntryIds"
CssClass="span-4 last"
            runat="server" Text="* EntryIds (Comma Seperated) :" />
        <ektronUI:TextField ID="uxEntryIds" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RecommendationManager recommendationManager = new RecommendationManager();
            long crossSellEntryId;
            long.TryParse(uxCrossSellEntryId.Text, out crossSellEntryId);
            string entryIds = uxEntryIds.Text;
            if (crossSellEntryId > 0 && entryIds != "")
            {
                CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
                EntryData entryData = catalogEntryManager.GetItem(crossSellEntryId);
                if (entryData != null && entryData.Id > 0)
                {
                    string[] entryIdArray = entryIds.Split(new char[]{','});
                    RecommendationItemData[] recommendationItemDataArray = new
RecommendationItemData[entryIdArray.Length];
                    for(int i = 0; i < entryIdArray.Length; i++ )
                    {
                        recommendationItemDataArray[i] = new RecommendationItemData();
                        recommendationItemDataArray[i].EntryId = long.Parse(entryIdArray
[i]);
                    }
```

```
                    recommendationManager.UpdateCrossSell(crossSellEntryId,
recommendationItemDataArray);
                    MessageUtilities.UpdateMessage(uxMessage, "EntryIds are Updated to
the CrossSell of EntryId " + crossSellEntryId.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
CrossSell Entry Id", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

UpdateUpSell

```
UpdateUpSell(System.Int64,System.Collections.Generic.IEnumerable)
```

Updates the UpSell.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * UpSell Entry ID
- * Entry IDs (Comma Separated)

Parameters

- `entryId`. ID of entry.
- `recommendations`. Enumerable of RecommendationItemData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUpSellEntryIdLabel" AssociatedControlID="uxUpSellEntryId"
```

```
CssClass="span-4 last"
            runat="server" Text="* UpSell EntryId :" />
        <ektronUI:TextField ID="uxUpSellEntryId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEntryIdsLabel" AssociatedControlID="uxEntryIds"
CssClass="span-4 last"
            runat="server" Text="* EntryIds (Comma Seperated) :" />
        <ektronUI:TextField ID="uxEntryIds" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RecommendationManager recommendationManager = new RecommendationManager();
            long upSellEntryId;
            long.TryParse(uxUpSellEntryId.Text, out upSellEntryId);
            string entryIds = uxEntryIds.Text;
            if (upSellEntryId > 0 && entryIds != "")
            {
                CatalogEntryManager catalogEntryManager = new CatalogEntryManager();
                EntryData entryData = catalogEntryManager.GetItem(upSellEntryId);
                if (entryData != null && entryData.Id > 0)
                {
                    string[] entryIdArray = entryIds.Split(new char[] { ',' });
                    RecommendationItemData[] recommendationItemDataArray = new
RecommendationItemData[entryIdArray.Length];
                    for (int i = 0; i < entryIdArray.Length; i++)
                    {
```

```
                        recommendationItemDataArray[i] = new RecommendationItemData();
                        recommendationItemDataArray[i].EntryId = long.Parse(entryIdArray
[i]);
                    }
                    recommendationManager.UpdateUpSell(upSellEntryId,
recommendationItemDataArray);
                    MessageUtilities.UpdateMessage(uxMessage, "EntryIds are Updated to
the UpSell of EntryId " + upSellEntryId.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid UpSell
Entry Id", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

RecommendationItemData

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `EntryLanguage`. Gets or sets the language of the entry.

  ```
  public long EntryLanguage { set; get; }
  ```

- `EntryType`. Gets or sets the entry type associated with this recommendation.

  ```
  public EkEnumeration.CatalogEntryType EntryType { set; get; }
  ```

- `Html`. Gets or sets the HTML of the recommendation.

  ```
  public string Html { set; get; }
  ```

- `Image`. Gets or sets the image associated with the recommendation.

  ```
  public string Image { set; get; }
  ```

- `ImageThumbnail`. Gets or sets the image thumbnail associated with the recommendation.

  ```
  public string ImageThumbnail { set; get; }
  ```

- `Media`

```
public Ektron.Cms.MediaGalleryData Media { set; get; }
```

- `Pricing`. The entry pricing, including currencies and quantity. Returns PricingData of the catalog entry.

```
public Ektron.Cms.Commerce.PricingData Pricing { set; get; }
```

- `Quicklink`. Gets or sets the quicklink of the recommendation.

```
public string Quicklink { set; get; }
```

- `RecommendationItemData()`

```
public RecommendationItemData()
```

- `RecommendationItemData(long, long, long, EkEnumeration.RecommendationType)`

```
public RecommendationItemData(long id, long entryId,
   long entryLanguage, EkEnumeration.RecommendationType
   recommendationType)
```

- `Summary`. Gets or sets the summary of the recommendation.

```
public string Summary { set; get; }
```

- `Title`. Gets or sets the title of the recommendation.

```
public string Title { set; get; }
```

- `Type`. Gets or sets the type of recommendation.

```
public EkEnumeration.RecommendationType Type { set; get; }
```

RecommendationType

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `CrossSell`

```
Public Const CrossSell
   As Ektron.Cms.Common.EkEnumeration.RecommendationType = 0
```

- `Upsell`

```
public Const Upsell
  As Ektron.Cms.Common.EkEnumeration.RecommendationType = 1
```

RegionManager

9.00 and higher

The class manages regions in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.RegionManager
```

Constructors

- `RegionManager()`
- `RegionManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Commerce.RegionData)
```

Adds a region to the CMS. The Region.Id property is populated with the new regions ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Name
- * Country ID
- * Code
- Enabled

Parameters

- `regionData`. RegionData object to add.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxRegionNameLabel" AssociatedControlID="uxRegionName"
CssClass="span-3 last" runat="server" Text="* Name:" />
        <ektronUI:TextField  ID="uxRegionName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionCountryIdLabel"
AssociatedControlID="uxRegionCountryId" CssClass="span-3 last" runat="server" Text="*
Country Id:" />
        <ektronUI:TextField  ID="uxRegionCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxRegionCodeLabel" AssociatedControlID="uxRegionCode"
CssClass="span-3 last" runat="server" Text="* Code:" />
        <ektronUI:TextField  ID="uxRegionCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionEnabledLabel" AssociatedControlID="uxRegionEnabled"
CssClass="span-3 last" runat="server" Text=" Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxRegionEnabled"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            RegionManager regionManager = new RegionManager();
            bool isEnabled = false;
            if (uxRegionEnabled.Checked)
            {
                isEnabled = true;
            }
            RegionData regionData = new RegionData()
            {
                Code = uxRegionCode.Text,
                CountryId = long.Parse(uxRegionCountryId.Text),
                Name = uxRegionName.Text,
                Enabled = isEnabled
            };

            regionData = regionManager.Add(regionData);
            if (regionData.Id > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Region Added with Id " +
regionData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Region adding process is
failed", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

CanDelete

```
CanDelete(System.Int64)
```

Checks to see whether a region can be deleted.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Region ID

Parameters

- `regionId`. ID of Region to delete.

Returns

True if the region can be deleted safely.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLabel" AssociatedControlID="uxRegionId"
CssClass="span-3 last" runat="server" Text="* Region Id:" />
        <ektronUI:TextField  ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="CanDelete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using Microsoft.Practices.EnterpriseLibrary.Validation;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            RegionManager regionManager = new RegionManager();
            long regionId = long.Parse(uxRegionId.Text);


            bool delete = regionManager.CanDelete(regionId);
            if (delete)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Region " + regionId + " can
be deleted", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Region " + regionId + "
cannot be deleted", Message.DisplayModes.Information);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Delete

```
Delete(System.Int64)
```

Deletes a region from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Region ID

Parameters

- `regionId`. ID of region to delete.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLabel" AssociatedControlID="uxRegionId"
CssClass="span-3 last" runat="server" Text="* Region Id:" />
        <ektronUI:TextField  ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            RegionManager regionManager = new RegionManager();
            long regionId = int.Parse(uxRegionId.Text);

            regionManager.Delete(regionId);
            MessageUtilities.UpdateMessage(uxMessage, "Region " + regionId + " is
deleted", Message.DisplayModes.Success );

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a region by region ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Region ID

Parameters

- `regionId`. ID of region to retrieve.

Returns

RegionData object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxRegionIDLabel" AssociatedControlID="uxRegionId"
CssClass="span-3 last" runat="server" Text="* Region Id :" />
        <ektronUI:TextField ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAlpha" runat="server"></asp:Literal>
    </li>
```

```
    <li class="clearfix">
        <asp:Literal ID="uxCultureCode" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxEnabled" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            RegionManager regionManager = new RegionManager();
            long regionId=long.Parse(uxRegionId.Text);
            RegionData regionData = regionManager.GetItem(regionId);

            if (regionData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for Region with ID "
+ regionData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxAlpha.Text = "Code : " + regionData.Code;
                uxCultureCode.Text = "Country Id : " + regionData.CountryId;
                uxName.Text = "Name : " + regionData.Name;
                uxEnabled.Text = "Enabled : " + regionData.Enabled;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Region "+regionId + " does
not exist. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

GetList

```
GetList(Ektron.Cms.Commerce.RegionCriteria)
```

Retrieves a list of regions.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Region Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter regions being retrieved.

Returns

List of regions meeting criteria filters.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxRegionPropertyLabel"
AssociatedControlID="uxRegionProperty" CssClass="span-4 last" runat="server"
Text="Region Property:" />
        <asp:DropDownList ID="uxRegionProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
            <asp:ListItem>Country Id </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCouponListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Name
                    </th>
                    <th>
                        Country Id
                    </th>
                    <th>
                        Code
                    </th>
                    <th>
                        Enabled
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("CountryId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Code")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Enabled")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            object Objectvalue;
            RegionCriteria criteria = new RegionCriteria();

            if (uxRegionProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(RegionProperty.Id,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxRegionProperty.SelectedItem.Text == "Name")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(RegionProperty.Name,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(RegionProperty.CountryId ,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            RegionManager regionManger = new RegionManager();
            List<RegionData> regionList = regionManger.GetList(criteria);

            uxRegionListView.Visible = true;
            uxRegionListView.DataSource = regionList;
            uxRegionListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
     }
```

Update

```
Update(Ektron.Cms.Commerce.RegionData)
```

Updates an existing region in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Name
- * Country ID
- * Code

Parameters

- `regionData`. RegionData object to save.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLabel" AssociatedControlID="uxRegionId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField  ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxRegionNameLabel" AssociatedControlID="uxRegionName"
CssClass="span-3 last" runat="server" Text="* Name:" />
        <ektronUI:TextField  ID="uxRegionName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionCountryIdLabel"
AssociatedControlID="uxRegionCountryId" CssClass="span-3 last" runat="server" Text="*
Country Id:" />
        <ektronUI:TextField  ID="uxRegionCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Label ID="uxRegionCodeLabel" AssociatedControlID="uxRegionCode"
CssClass="span-3 last" runat="server" Text="* Code:" />
        <ektronUI:TextField  ID="uxRegionCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionEnabledLabel" AssociatedControlID="uxRegionEnabled"
CssClass="span-3 last" runat="server" Text=" Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxRegionEnabled"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            RegionManager regionManager = new RegionManager();
            bool enabled = false;
            if (uxRegionEnabled.Checked)
            {
                enabled = true ;
            }
            long regionId=long.Parse(uxRegionId.Text);
            RegionData regionData = regionManager.GetItem(regionId);
            if (regionData != null)
            {
                regionData.Code  = uxRegionCode.Text;
                regionData.CountryId = long.Parse(uxRegionCountryId.Text);
                regionData.Name = uxRegionName.Text;
                regionData.Enabled = enabled;
                regionData = regionManager.Update(regionData);

                MessageUtilities.UpdateMessage(uxMessage, "Region updated with Id " +
regionData.Id, Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Data Classes

RegionCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `RegionCriteria()`

  ```
  public RegionCriteria()
  ```

- `RegionCriteria(Ektron.Cms.Commerce.RegionProperty, EkEnumeration.OrderByDirection)`

  ```
  public RegionCriteria(Ektron.Cms.Commerce.RegionProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `RegionProperty` are:

  - AlphaCode
  - CountryId
  - Id
  - IsEnabled
  - Name

RegionData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `Code`. Gets or sets the code of the region. This is typically an abbreviation for the name of the region. For example, if you add New Hampshire as a region, you would enter "NH" for this property. Each region must have its own unique code. You cannot have more than one region with the same code. This property is required when creating an region data object.

  ```
  public string Code { set; get; }
  ```

- `CountryId`. Gets or sets the ID of the country to which the region belongs. For example, if you are adding a region to the United States, you would enter "840" for this property. This property is required when creating an region data object.

  ```
  public long CountryId { set; get; }
  ```

- `Enabled`. Gets or sets whether or not the region is enabled for use.

```
public bool Enabled { set; get; }
```

- `Id`. Gets or sets the ID of the region.

```
public long Id { set; get; }
```

- `Name`. Gets or sets the name of the region. For example, if the region is a US state, you might enter "New Hampshire" for this property. This property is required when creating an region data object.

```
public string Name { set; get; }
```

- `RegionData()`

```
public RegionData()
```

- `RegionData(string, long, string, bool)`

```
public RegionData(string name, long countryId, string code, bool enabled)
```

- `ToString()`. Returns Name of Region.

```
public override string ToString()
```

- `Validate()`. Validates the RegionData object is valid for saving. Returns a ValidationResult object with any validation errors.

```
public ValidationResults Validate()
```

ShippingMethodManager

9.00 and higher

The class manages shipping methods in CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.ShippingMethodManager
```

Constructors

- `ShippingMethodManager()`
- `ShippingMethodManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 732
- *GetItem* on page 734
- *GetList* on page 736
- *Update* on page 739

Add

```
Add(Ektron.Cms.Commerce.ShippingMethodData)
```

Adds a shipping method to the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Shipping Method Name
- * Provider Service
- * Display Order
- * Is Active

Parameters

- `shippingMethodData`. ShippingMethodData object to add.

Returns

ShippingMethodData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-5
last" runat="server" Text="* Shipping Method Name:" />
        <ektronUI:TextField ID="uxName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxProviderServiceLabel"
AssociatedControlID="uxProviderService" CssClass="span-5 last" runat="server" Text="*
ProviderService:" />
        <ektronUI:TextField ID="uxProviderService" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
        </li>
        <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxDisplayOrderLabel" AssociatedControlID="uxDisplayOrder"
CssClass="span-5 last" runat="server" Text="* DisplayOrder:" />
        <ektronUI:TextField ID="uxDisplayOrder" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
        </li>
        <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsActiveLabel" AssociatedControlID="uxIsActive"
CssClass="span-5 last" runat="server" Text="* IsActive: " />
        <asp:checkbox ID="uxIsActive" runat="server" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
```

```
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ShippingMethodManager shippingMethodManager = new ShippingMethodManager();
            ShippingMethodData shippingMethodData=new ShippingMethodData();
            shippingMethodData.Name = uxName.Text;
            shippingMethodData.ProviderService =uxProviderService.Text;
            shippingMethodData.DisplayOrder = int.Parse(uxDisplayOrder.Text);
            shippingMethodData.IsActive = uxIsActive.Checked;

            shippingMethodData = shippingMethodManager.Add(shippingMethodData);
            MessageUtilities.UpdateMessage(uxMessage, "shippingMethodData is added with
Id " + shippingMethodData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a shipping method from CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `shippingMethodId`. ID of the shipping method to delete.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingIdLabel" AssociatedControlID="uxShippingId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxShippingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxShippingId.Text);

            ShippingMethodManager shippingMethodManager = new ShippingMethodManager();
            shippingMethodManager.Delete(ID);
            MessageUtilities.UpdateMessage(uxMessage, "ShippingMethodData with Id " +
uxShippingId.Text + " has been deleted", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the single shipping method by shipping method ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `shippingMethodId`. ID of the shipping method to retrieved.

Returns

ShippingMethodData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingIdLabel" AssociatedControlID="uxShippingId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxShippingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
    </ol>
    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxProviderService" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDisplayOrder" runat="server"></asp:Literal>
    </li>
```

```
    <li class="clearfix">
        <asp:Literal ID="uxIsActive" runat="server"></asp:Literal>
    </li>
 </ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long shippingMethodId = long.Parse(uxShippingId.Text);

            ShippingMethodManager shippingMethodManager = new ShippingMethodManager();
            ShippingMethodData shippingMethodData = new ShippingMethodData();
            shippingMethodData = shippingMethodManager.GetItem(shippingMethodId);
            if (shippingMethodData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for shipping method
Id " + shippingMethodData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxName.Text = "Name: " + shippingMethodData.Name;
                uxProviderService.Text = "ProviderService: " +
shippingMethodData.ProviderService;
                uxDisplayOrder.Text = "DisplayOrder: " +
shippingMethodData.DisplayOrder;
                uxIsActive.Text = "IsActive: " + shippingMethodData.IsActive;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either shipping method with
ID " + shippingMethodId + " does not exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.ShippingMethodCriteria)
```

Retrieves a list of shipping methods.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Shipping Method Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter shipping methods being retrieved.

Returns

List of shipping methods meeting criteria filters.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingMethodPropertyLabel"
AssociatedControlID="uxShippingMethodProperty" CssClass="span-5 last" runat="server"
Text="* ShippingMethod Property:" />
        <asp:DropDownList ID="uxShippingMethodProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
            <asp:ListItem>ProviderService</asp:ListItem>
            <asp:ListItem>DisplayOrder</asp:ListItem>
            <asp:ListItem>IsActive</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
```

```
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxShippingMethodDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >No records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>Id</th>
                    <th>Name</th>
                    <th>DisplayOrder</th>
                    <th>ProviderService</th>
                    <th>IsActive</th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("DisplayOrder")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ProviderService")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsActive")%>
            </td>

      </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
```

```
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            ShippingMethodManager shippingMethodManager = new ShippingMethodManager();
            ShippingMethodCriteria criteria = new ShippingMethodCriteria();

            if (uxShippingMethodProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(ShippingMethodProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxShippingMethodProperty.SelectedItem.Text == "Name")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(ShippingMethodProperty.Name,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxShippingMethodProperty.SelectedItem.Text == "DisplayOrder")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(ShippingMethodProperty.DisplayOrder,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxShippingMethodProperty.SelectedItem.Text == "ProviderService")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(ShippingMethodProperty.ProviderService,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(ShippingMethodProperty.IsActive,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            List<ShippingMethodData> ShippingMethodDataList =
shippingMethodManager.GetList(criteria);

            uxShippingMethodDataListView.Visible = true;
            uxShippingMethodDataListView.DataSource = ShippingMethodDataList;
            uxShippingMethodDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
```

```
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.ShippingMethodData)
```

Updates an existing shipping method in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Shipping Method Name
- * Provider Service
- * Display Order
- * Is Active

Parameters

- `shippingMethodData`. ShippingMethodData object to save.

Returns

ShippingMethodData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingIdLabel" AssociatedControlID="uxShippingId"
CssClass="span-5 last"  runat="server" Text=" Id:" />
        <ektronUI:TextField ID="uxShippingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-5
last" runat="server" Text="* Shipping Method Name:" />
        <ektronUI:TextField ID="uxName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxProviderServiceLabel"
```

```
AssociatedControlID="uxProviderService" CssClass="span-5 last" runat="server" Text="*
ProviderService:" />
        <ektronUI:TextField ID="uxProviderService" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
        </li>
        <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxDisplayOrderLabel" AssociatedControlID="uxDisplayOrder"
CssClass="span-5 last" runat="server" Text="* DisplayOrder:" />
        <ektronUI:TextField ID="uxDisplayOrder" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
        </li>
        <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsActiveLabel" AssociatedControlID="uxIsActive"
CssClass="span-5 last" runat="server" Text="* IsActive: " />
            <asp:checkbox ID="uxIsActive" runat="server" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long shppingMethodId = long.Parse(uxShippingId.Text);
            ShippingMethodManager shippingMethodManager = new ShippingMethodManager();
            ShippingMethodData shippingMethodData = new ShippingMethodData();
            shippingMethodData = shippingMethodManager.GetItem(shppingMethodId);
            if (shppingMethodId > 0 && shippingMethodData != null)
            {
                shippingMethodData.Name = uxName.Text;
                shippingMethodData.DisplayOrder = int.Parse(uxDisplayOrder.Text);
                shippingMethodData.ProviderService = uxProviderService.Text;
```

```
                shippingMethodData.IsActive = uxIsActive.Checked;
                shippingMethodData = shippingMethodManager.Update(shippingMethodData);
                MessageUtilities.UpdateMessage(uxMessage, "Shipping Method Data has been
updated with Id " + uxShippingId.Text, Message.DisplayModes.Success);

            }
            else
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Shipping
Method ID.", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

ShippingMethodCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `ShippingMethodCriteria()`

  ```
  public ShippingMethodCriteria()
  ```

- `ShippingMethodCriteria (Ektron.Cms.Commerce.ShippingMethodProperty, EkEnumeration.OrderByDirection)`

  ```
  public ShippingMethodCriteria(Ektron.Cms.Commerce.ShippingMethodProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `ShippingMethodProperty` are:

  - DisplayOrder
  - Id
  - IsActive
  - Name
  - ProviderSearch

ShippingMethodData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `DisplayOrder`. Gets or sets the display order for this shipping method.

  ```
  public int DisplayOrder { set; get; }
  ```

- `IsActive`. Gets or sets whethe ror not the Shipping Method is Active.

  ```
  public bool IsActive { set; get; }
  ```

- `ShippingMethodsData()`

  ```
  public ShippingMethodData()
  ```

TaxClassManager

9.00 and higher

The class manages tax class in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.TaxClassManager
```

Constructors

- `TaxClassManager()`
- `TaxClassManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 745
- *GetItem* on page 746
- *GetList* on page 748
- *IsUsed* on page 750
- *Update* on page 752

Add

```
Add(Ektron.Cms.Commerce.TaxClassData)
```

Adds a tax class to the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Tax Class Name

Parameters

- `taxClassData`. TaxClassData object to add.

.aspx code snippet

```
<ol class="formFields">
<li class="clearfix">
    <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-6 last"
runat="server" Text="* Tax Class Name:" />
    <ektronUI:TextField ID="uxName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
</li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TaxClassManager taxClassManager = new TaxClassManager();
            TaxClassData taxClassData = new TaxClassData();
            if (uxName.Text.ToString().Trim() != "")
            {
                taxClassData.Name = uxName.Text;
                taxClassData = taxClassManager.Add(taxClassData);
```

```
                MessageUtilities.UpdateMessage(uxMessage, "TaxClassData is added with Id
" + taxClassData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Tax Class Name should not be
empty ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a tax class from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of TaxClass to delete.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxClassIdLabel" AssociatedControlID="uxTaxClassId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTaxClassId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxTaxClassId.Text);
            TaxClassManager taxClassManager = new TaxClassManager();
            taxClassManager.Delete(ID);
            MessageUtilities.UpdateMessage(uxMessage, "TaxClassData with Id " +
uxTaxClassId.Text + " is deleted", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a tax class by ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of taxClass to retrieve.

Returns

TaxClassData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxClassIdLabel" AssociatedControlID="uxTaxClassId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTaxClassId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
    </ol>
    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
   </ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxClassId = long.Parse(uxTaxClassId.Text);

            TaxClassManager taxClassManager = new TaxClassManager();
            TaxClassData taxClassData = new TaxClassData();
            taxClassData = taxClassManager.GetItem(taxClassId);
            if (taxClassData != null)
            {
```

```
                MessageUtilities.UpdateMessage(uxMessage, "Details for Tax Class Id " +
taxClassData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxName.Text = "Name : " + taxClassData.Name;

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either Tax class data with ID
" + taxClassId + " does not exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.TaxClassCriteria)
```

Retrieves a list of tax classes.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Tax Class Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter taxClass being retrieved.

Returns

List of TaxClassData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxClassPropertyLabel" AssociatedControlID="uxTaxClass"
```

```
CssClass="span-5 last" runat="server" Text="* Tax Class Property:" />
        <asp:DropDownList ID="uxTaxClass" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxtaxClassDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >No records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>Id</th>
                    <th>Name</th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            TaxClassManager taxClassManager = new TaxClassManager();
            TaxClassCriteria criteria = new TaxClassCriteria();
            if (uxTaxClass.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(TaxClassProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxClassProperty.Name,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            List<TaxClassData> taxClassDataList = taxClassManager.GetList(criteria);
            uxtaxClassDataListView.Visible = true;
            uxtaxClassDataListView.DataSource = taxClassDataList;
            uxtaxClassDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IsUsed

```
IsUsed(System.Int64)
```

Returns whether a tax class is used in a product.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of tax class.

Returns

Returns true if the tax class is used in a product.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxClassIdLabel" AssociatedControlID="uxTaxClassId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTaxClassId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="IsUsed"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxClassId = long.Parse(uxTaxClassId.Text);
            TaxClassManager taxClassManager = new TaxClassManager();
            bool result = taxClassManager.IsUsed(taxClassId);
            if (result)
               MessageUtilities.UpdateMessage(uxMessage, "TaxClassData has been used",
Message.DisplayModes.Success);
```

```
            else
                MessageUtilities.UpdateMessage(uxMessage, "TaxClassData hasn't been
used", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.TaxClassData)
```

Updates an existing tax class in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Tax Class Name

Parameters

- `taxClassData`. TaxClassData object to save.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxClassIdLabel" AssociatedControlID="uxTaxClassId"
CssClass="span-6 last" runat="server" Text=" Id:" />
        <ektronUI:TextField ID="uxTaxClassId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-6
last" runat="server" Text="* Tax Class Name:" />
        <ektronUI:TextField ID="uxName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
```

```
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxClassId = long.Parse(uxTaxClassId.Text);
            TaxClassManager taxClassManager = new TaxClassManager();
            TaxClassData taxClassData = new TaxClassData();
            taxClassData = taxClassManager.GetItem(taxClassId);
            if (taxClassId > 0 && taxClassData != null)
            {
                if (uxName.Text.ToString().Trim() != "")
                {
                    taxClassData.Name = uxName.Text;
                    taxClassData = taxClassManager.Update(taxClassData);
                    MessageUtilities.UpdateMessage(uxMessage, "TaxClassData has been
updated with Id " + uxTaxClassId.Text, Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Tax Class Name should not
be empty ", Message.DisplayModes.Error);
                }

            }
            else
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid TaxClass
ID.", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

TaxClassCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `TaxClassCriteria()`

  ```
  public TaxClassCriteria()
  ```

- `TaxClassCriteria(Ektron.Cms.Commerce.TaxClassProperty, EkEnumeration.OrderByDirection)`

  ```
  public TaxClassCriteria(Ektron.Cms.Commerce.TaxClassProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `TaxClassProperty` are:

  - Id
  - Name

TaxClassData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `Id`. Gets or sets the name of the tax class.

  ```
  public long Id { set; get; }
  ```

- `Name`. Gets or sets the name of the tax class.

  ```
  public string Name { set; get; }
  ```

- `TaxClassData()`

  ```
  public TaxClassData()
  ```

- `TaxClassData(string)`

  ```
  public TaxClassData(string name)
  ```

- `TaxClassData(string, long)`

  ```
  public TaxClassData(string name, long id)
  ```

TaxRateManager

9.00 and higher

The class manages commerce tax rates in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.TaxRateManager
```

Constructors

- `TaxRateManager()`
- `TaxRateManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Commerce.TaxRateData)
```

Adds a tax rate to the CMS. The Tax.Id property is populated with the new Tax's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Rate
- * Tax Class ID
- * Type ID
- * Type Item ID

Parameters

- `taxRateData`. TaxRateData object to add.

.aspx code snippet

```
<ol class="formFields">
<li class="clearfix">
    <ektronUI:Label ID="uxRateLabel" AssociatedControlID="uxRate" CssClass="span-4 last"
        runat="server" Text="* Rate:" />
    <ektronUI:TextField ID="uxRate" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
        Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Label ID="uxTaxClassIdLabel" AssociatedControlID="uxTaxClassId"
CssClass="span-4 last"
        runat="server" Text="* TaxClassId:" />
    <ektronUI:TextField ID="uxTaxClassId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
        Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Label ID="uxTypeIdLabel" AssociatedControlID="uxTypeId" CssClass="span-4
last"
        runat="server" Text="* TypeId:" />
    <ektronUI:TextField ID="uxTypeId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
        Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Label ID="uxTypeItemIdLabel" AssociatedControlID="uxTypeItemId"
CssClass="span-4 last"
```

```
        runat="server" Text="* TypeItemId:" />
    <ektronUI:TextField ID="uxTypeItemId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
        Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            TaxRateManager taxRateManager = new TaxRateManager();
            TaxRateData taxRateData = new TaxRateData();
            taxRateData.Rate = decimal.Parse(uxRate.Text);
            taxRateData.TaxClassId = long.Parse(uxTaxClassId.Text);
            taxRateData.TypeId = long.Parse(uxTypeId.Text);
            taxRateData.TypeItemId = long.Parse(uxTypeItemId.Text);

            taxRateData = taxRateManager.Add(taxRateData);
            MessageUtilities.UpdateMessage(uxMessage, "TaxRateData is added with Id " +
taxRateData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a tax rate from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `taxRateId`. ID of tax to delete.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxRateIdLabel" AssociatedControlID="uxTaxRateId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTaxRateId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            TaxRateManager taxRateManager = new TaxRateManager();
            long ID = long.Parse(uxTaxRateId.Text);
            TaxRateData taxRateData = new TaxRateData();
            taxRateData = taxRateManager.GetItem(ID);
            if (ID > 0 && taxRateData != null)
            {
```

```
                taxRateManager.Delete(ID);
                MessageUtilities.UpdateMessage(uxMessage, "TaxRateData with Id " +
uxTaxRateId.Text + " is deleted", Message.DisplayModes.Success);
            }
            else
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid TaxRate
ID.", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeleteByCountry

```
DeleteByCountry(System.Int32)
```

Deletes all rates associated with a country.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `countryId`. ID of country from which to delete rates.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIdLabel" AssociatedControlID="uxCountryId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="DeleteByCountry"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            TaxRateManager taxRateManager = new TaxRateManager();
            int ID = int.Parse(uxCountryId.Text);
            taxRateManager.DeleteByCountry(ID);
            MessageUtilities.UpdateMessage(uxMessage, "TaxRateData with countryId " +
uxCountryId.Text + " is deleted", Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeleteByRegion

```
DeleteByRegion(System.Int64)
```

Deletes all rates associated with a region

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `regionId`. ID of region from which to delete rates.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLabel" AssociatedControlID="uxRegionId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="DeleteByRegion"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            TaxRateManager taxRateManager = new TaxRateManager();
            long ID = long.Parse(uxRegionId.Text);
            taxRateManager.DeleteByRegion(ID);
            MessageUtilities.UpdateMessage(uxMessage, "TaxRateData with regionId " +
uxRegionId.Text + " is deleted", Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetApplicableTaxRate

```
GetApplicableTaxRate
(Ektron.Cms.Commerce.AddressData,Ektron.Cms.Commerce.AddressData,System.Int64)
```

Returns the applicable tax rate for the supplied addresses and tax class.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Tax Class ID

Shipping From Address

- * Shipping From Address Name
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID

Shipping To Address

- * Shipping To Address Name
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID

Parameters

- `shipFromAddress`. Address where the products are coming from.
- `shipToAddress`. Address where products are going.
- `taxClassId`. The taxClass for which to retrieve a rate.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxClassIdLabel" AssociatedControlID="uxTaxClass"
CssClass="span-6 last" runat="server" Text="* TaxClassId: " />
        <ektronUI:TextField ID="uxTaxClass" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <b>Shipping From Address</b>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromAddressNameLabel"
AssociatedControlID="uxShippingFromAddressName" CssClass="span-6 last"
```

```
            runat="server" Text="* Shipping From AddressName :" />
        <ektronUI:TextField ID="uxShippingFromAddressName" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromAddressLine1Label"
AssociatedControlID="uxShippingFromAddressLine1" CssClass="span-6 last"
            runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxShippingFromAddressLine1" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromCityLabel"
AssociatedControlID="uxShippingFromCity" CssClass="span-6 last"
            runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxShippingFromCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromRegionIdLabel"
AssociatedControlID="uxShippingFromRegionId" CssClass="span-6 last"
            runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxShippingFromRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromPostalCodeLabel"
AssociatedControlID="uxShippingFromPostalCode" CssClass="span-6 last"
            runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxShippingFromPostalCode" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromCountryIdLabel"
AssociatedControlID="uxShippingFromCountryId" CssClass="span-6 last"
            runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxShippingFromCountryId" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <b>Shipping To Address</b>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToAddressNameLabel"
AssociatedControlID="uxShippingToAddressName" CssClass="span-6 last"
            runat="server" Text="* Shipping To AddressName :" />
        <ektronUI:TextField ID="uxShippingToAddressName" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToAddressLine1Label"
AssociatedControlID="uxShippingToAddressLine1" CssClass="span-6 last"
            runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxShippingToAddressLine1" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToCityLabel"
AssociatedControlID="uxShippingToCity" CssClass="span-6 last"
            runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxShippingToCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToRegionIdLabel"
AssociatedControlID="uxShippingToRegionId" CssClass="span-6 last"
            runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxShippingToRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToPostalCodeLabel"
AssociatedControlID="uxShippingToPostalCode" CssClass="span-6 last"
            runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxShippingToPostalCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToCountryIdLabel"
AssociatedControlID="uxShippingToCountryId" CssClass="span-6 last"
            runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxShippingToCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetApplicableTaxRate"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxRate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTaxClassId" runat="server"></asp:Literal>
    </li>
        <li class="clearfix">
        <asp:Literal ID="uxTypeId" runat="server"></asp:Literal>
    </li>
        <li class="clearfix">
        <asp:Literal ID="uxTypeItemId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TaxRateManager taxRateManager = new TaxRateManager();
            TaxRateData taxRateData = new TaxRateData();
            long taxClassId = long.Parse(uxTaxClassId.Text);

            //ShippingFromAddress Information
            AddressData shippingFromAddress = new AddressData();
            shippingFromAddress.Name = uxShippingFromAddressName.Text;
            shippingFromAddress.AddressLine1 = uxShippingFromAddressLine1.Text;
            shippingFromAddress.City = uxShippingFromCity.Text;
            shippingFromAddress.Region = new RegionData { Id = long.Parse
(uxShippingFromRegionId.Text) };
            shippingFromAddress.PostalCode = uxShippingFromPostalCode.Text;
            shippingFromAddress.Country = new CountryData { Id = int.Parse
(uxShippingFromCountryId.Text) };

            //ShippingToAddress Information
            AddressData shippingToAddress = new AddressData();
            shippingToAddress.Name = uxShippingToAddressName.Text;
            shippingToAddress.AddressLine1 = uxShippingToAddressLine1.Text;
            shippingToAddress.City = uxShippingToCity.Text;
            shippingToAddress.Region = new RegionData { Id = long.Parse
(uxShippingToRegionId.Text) };
            shippingToAddress.PostalCode = uxShippingToPostalCode.Text;
            shippingToAddress.Country = new CountryData { Id = int.Parse
(uxShippingToCountryId.Text) };

            taxRateData = taxRateManager.GetApplicableTaxRate(shippingFromAddress,
shippingToAddress,taxClassId);
            if (taxRateData != null)
            {

                MessageUtilities.UpdateMessage(uxMessage, "Details for Tax rate data Id
" + taxRateData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxRate.Text = "Rate : " + taxRateData.Rate;
                uxTaxClassId.Text = "TaxClassId : " + taxRateData.TaxClassId;
                uxTypeId.Text = "TypeId : " + taxRateData.TypeId;
                uxTypeItemId.Text = "TypeItemId : " + taxRateData.TypeItemId;
```

```
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either Tax rate does not
exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetApplicableTaxRateList

```
GetApplicableTaxRateList
(Ektron.Cms.Commerce.AddressData,Ektron.Cms.Commerce.AddressData)
```

Returns a list of applicable tax rates for the supplied addresses. There is at most one rate for each available tax class for the specified addresses.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

Shipping From Address

- * Shipping From Address Name
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID

Shipping To Address

- * Shipping To Address Name
- * Address Line 1
- * City
- * Region ID

- * Postal Code
- * Country ID

Parameters

- `shipFromAddress`. Address where the products are coming from.
- `shipToAddress`. Address where products are going.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <b>Shipping From Address</b>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromAddressNameLabel"
AssociatedControlID="uxShippingFromAddressName" CssClass="span-5 last"
            runat="server" Text="* Shipping From AddressName :" />
        <ektronUI:TextField ID="uxShippingFromAddressName" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromAddressLine1Label"
AssociatedControlID="uxShippingFromAddressLine1" CssClass="span-5 last"
            runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxShippingFromAddressLine1" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromCityLabel"
AssociatedControlID="uxShippingFromCity" CssClass="span-5 last"
            runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxShippingFromCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromRegionIdLabel"
AssociatedControlID="uxShippingFromRegionId" CssClass="span-5 last"
            runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxShippingFromRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromPostalCodeLabel"
AssociatedControlID="uxShippingFromPostalCode" CssClass="span-5 last"
            runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxShippingFromPostalCode" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingFromCountryIdLabel"
AssociatedControlID="uxShippingFromCountryId" CssClass="span-5 last"
            runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxShippingFromCountryId" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <b>Shipping To Address</b>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToAddressNameLabel"
AssociatedControlID="uxShippingToAddressName" CssClass="span-5 last"
            runat="server" Text="* Shipping To AddressName :" />
        <ektronUI:TextField ID="uxShippingToAddressName" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToAddressLine1Label"
AssociatedControlID="uxShippingToAddressLine1" CssClass="span-5 last"
            runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxShippingToAddressLine1" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToCityLabel"
AssociatedControlID="uxShippingToCity" CssClass="span-5 last"
            runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxShippingToCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToRegionIdLabel"
AssociatedControlID="uxShippingToRegionId" CssClass="span-5 last"
            runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxShippingToRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToPostalCodeLabel"
AssociatedControlID="uxShippingToPostalCode" CssClass="span-5 last"
            runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxShippingToPostalCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxShippingToCountryIdLabel"
AssociatedControlID="uxShippingToCountryId" CssClass="span-5 last"
            runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxShippingToCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetApplicableTaxRateList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>
<ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
    <asp:ListView ID="uxtaxRateDataListView" runat="server"
```

```
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                <th>Id</th>
                <th>Rate</th>
                <th>TaxClassId</th>
                <th>TypeId</th>
                <th>TypeItemId</th>
                </tr>
            </thead>
            <tbody>
            <asp:PlaceHolder ID="aspItemPlaceholder" runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("Rate")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TaxClassId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TypeId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TypeItemId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            TaxRateManager taxRateManager = new TaxRateManager();
            TaxRateData taxRateData = new TaxRateData();

            //ShippingFromAddress Information
            AddressData shippingFromAddress = new AddressData();
            shippingFromAddress.Name = uxShippingFromAddressName.Text;
            shippingFromAddress.AddressLine1 = uxShippingFromAddressLine1.Text;
            shippingFromAddress.City = uxShippingFromCity.Text;
            shippingFromAddress.Region = new RegionData { Id = long.Parse
(uxShippingFromRegionId.Text) };
            shippingFromAddress.PostalCode = uxShippingFromPostalCode.Text;
            shippingFromAddress.Country = new CountryData { Id = int.Parse
(uxShippingFromCountryId.Text) };

            //ShippingToAddress Information
            AddressData shippingToAddress = new AddressData();
            shippingToAddress.Name = uxShippingToAddressName.Text;
            shippingToAddress.AddressLine1 = uxShippingToAddressLine1.Text;
            shippingToAddress.City = uxShippingToCity.Text;
            shippingToAddress.Region = new RegionData { Id = long.Parse
(uxShippingToRegionId.Text) };
            shippingToAddress.PostalCode = uxShippingToPostalCode.Text;
            shippingToAddress.Country = new CountryData { Id = int.Parse
(uxShippingToCountryId.Text) };

            List<TaxRateData> taxRateDataList = taxRateManager.GetApplicableTaxRateList
(shippingFromAddress, shippingToAddress);

            uxtaxRateDataListView.Visible = true;
            uxtaxRateDataListView.DataSource = taxRateDataList;
            uxtaxRateDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Gets a Tax Rate based upon Tax Rate ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `taxRateId`. ID of tax rate to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxRateIdLabel" AssociatedControlID="uxTaxRateId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTaxRateId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
    </ol>
   <ol class="formFields">
        <li class="clearfix">
            <ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
        </li>
        <li class="clearfix">
            <asp:Literal ID="uxRate" runat="server"></asp:Literal>
        </li>
        <li class="clearfix">
            <asp:Literal ID="uxTaxClassId" runat="server"></asp:Literal>
        </li>
            <li class="clearfix">
            <asp:Literal ID="uxTypeId" runat="server"></asp:Literal>
        </li>
            <li class="clearfix">
            <asp:Literal ID="uxTypeItemId" runat="server"></asp:Literal>
        </li>
   </ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxRateId = long.Parse(uxTaxRateId.Text);

            TaxRateManager taxRateManager = new TaxRateManager();
            TaxRateData taxRateData = new TaxRateData();
            taxRateData = taxRateManager.GetItem(taxRateId);
            if (taxRateData != null)
            {

                MessageUtilities.UpdateMessage(uxMessage, "Details for Tax rate data Id
" + taxRateData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxRate.Text = "Rate : " + taxRateData.Rate;
                uxTaxClassId.Text = "TaxClassId : " + taxRateData.TaxClassId;
                uxTypeId.Text = "TypeId : " + taxRateData.TypeId;
                uxTypeItemId.Text = "TypeItemId : " + taxRateData.TypeItemId;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either Tax rate with Id " +
taxRateId + " does not exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.TaxRateCriteria)
```

Gets a list of rates based upon the supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Tax Rate Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter and sort tax rates.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxRatePropertyLabel" AssociatedControlID="uxTaxRate"
CssClass="span-5 last" runat="server" Text="* TaxRate Property:" />
        <asp:DropDownList ID="uxTaxRate" runat="server">
            <asp:ListItem>RateId</asp:ListItem>
            <asp:ListItem>Rate</asp:ListItem>
            <asp:ListItem>TaxClassId</asp:ListItem>
            <asp:ListItem>CountryId</asp:ListItem>
            <asp:ListItem>RegionId</asp:ListItem>
            <asp:ListItem>PostalCode</asp:ListItem>
            <asp:ListItem>TaxTypeId</asp:ListItem>
            <asp:ListItem>TaxTypeItemId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
<asp:View ID="uxViewMessage" runat="server">
<asp:ListView ID="uxtaxRateDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
<EmptyDataTemplate >Criteria didn't match the records.</EmptyDataTemplate>
<LayoutTemplate>
    <table class="devsite-api-method">
        <thead>
            <tr>
            <th>Id</th>
            <th>Rate</th>
            <th>TaxClassId</th>
            <th>TypeId</th>
            <th>TypeItemId</th>
            </tr>
```

```
        </thead>
        <tbody>
          <asp:PlaceHolder ID="aspItemPlaceholder" runat="server">
          </asp:PlaceHolder>
        </tbody>
    </table>
</LayoutTemplate>
<ItemTemplate>
    <tr>
        <td class="devsite-method">
            <%# Eval("Id")%>
        </td>
            <td class="devsite-method">
            <%# Eval("Rate")%>
        </td>
        <td class="devsite-method">
            <%# Eval("TaxClassId")%>
        </td>
        <td class="devsite-method">
            <%# Eval("TypeId")%>
        </td>
        <td class="devsite-method">
            <%# Eval("TypeItemId")%>
        </td>
    </tr>
</ItemTemplate>
</asp:ListView>
</asp:View>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            TaxRateManager taxRateManager = new TaxRateManager();
            TaxRateCriteria criteria = new TaxRateCriteria();

            if (uxTaxRate.SelectedItem.Text == "RateId")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(TaxRateProperty.RateId,
```

```
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxTaxRate.SelectedItem.Text == "Rate")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxRateProperty.Rate, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else if (uxTaxRate.SelectedItem.Text == "TaxClassId")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxRateProperty.TaxClassId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxTaxRate.SelectedItem.Text == "CountryId")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxRateProperty.CountryId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxTaxRate.SelectedItem.Text == "RegionId")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxRateProperty.RegionId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxTaxRate.SelectedItem.Text == "PostalCode")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxRateProperty.PostalCode,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxTaxRate.SelectedItem.Text == "TaxTypeId")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxRateProperty.TaxTypeId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxRateProperty.TaxTypeItemId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            List<TaxRateData> taxRateDataList = taxRateManager.GetList(criteria);

            uxtaxRateDataListView.Visible = true;
            uxtaxRateDataListView.DataSource = taxRateDataList;
            uxtaxRateDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.TaxRateData)
```

Updates an existing tax rate in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- ID
- * Rate
- * Tax Class ID
- * Type ID
- * Type Item ID

Parameters

- `taxRateData`. TaxRateData object to save.

.aspx code snippet

```
<ol class="formFields">
<li class="clearfix">
    <ektronUI:Label ID="uxTaxRateIdLabel" AssociatedControlID="uxTaxRateId"
CssClass="span-4 last" runat="server" Text=" Id:" />
    <ektronUI:TextField ID="uxTaxRateId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
</li>
<li class="clearfix">
<ektronUI:Label ID="uxRateLabel" AssociatedControlID="uxRate" CssClass="span-4 last"
    runat="server" Text="* Rate:" />
<ektronUI:TextField ID="uxRate" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
    Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
<ektronUI:Label ID="uxTaxClassIdLabel" AssociatedControlID="uxTaxClassId"
CssClass="span-4 last"
    runat="server" Text="* TaxClassId:" />
<ektronUI:TextField ID="uxTaxClassId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
```

```
    Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
<ektronUI:Label ID="uxTypeIdLabel" AssociatedControlID="uxTypeId" CssClass="span-4 last"
    runat="server" Text="* TypeId:" />
<ektronUI:TextField ID="uxTypeId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
    Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
<ektronUI:Label ID="uxTypeItemIdLabel" AssociatedControlID="uxTypeItemId"
CssClass="span-4 last"
    runat="server" Text="* TypeItemId:" />
<ektronUI:TextField ID="uxTypeItemId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
    Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxRateId = long.Parse(uxTaxRateId.Text);
            TaxRateManager taxRateManager = new TaxRateManager();
            TaxRateData taxRateData = new TaxRateData();
            taxRateData = taxRateManager.GetItem(taxRateId);
            if (taxRateId > 0 && taxRateData != null)
            {
                taxRateData.Rate = decimal.Parse(uxRate.Text);
```

```
                taxRateData.TaxClassId = long.Parse(uxTaxClassId.Text);
                taxRateData.TypeId = long.Parse(uxTypeId.Text);
                taxRateData.TypeItemId = long.Parse(uxTypeItemId.Text);
                taxRateData = taxRateManager.Update(taxRateData);
                MessageUtilities.UpdateMessage(uxMessage, "TaxRateData Updated with Id "
+ taxRateData.Id, Message.DisplayModes.Success);

            }
            else
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid TaxRate
ID.", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

TaxRateCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `TaxRateCriteria()`

  ```
  public TaxRateCriteria()
  ```

- `TaxRateCriteria(Ektron.Cms.Commerce.TaxRateProperty, EkEnumeration.OrderByDirection)`

  ```
  public TaxRateCriteria(Ektron.Cms.Commerce.TaxRateProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `TaxRateProperty` are:

  - Country
  - CountryId
  - PostalCode
  - Rate
  - RateId
  - Region
  - RegionId
  - TaxClassId

- TaxClassName
- TaxTypeId
- TaxTypeItemId
- TaxTypeName
- TaxTypePreference

TaxRateData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `Id`. Gets or sets the ID of the TaxRate.

  ```
  public long Id { set; get; }
  ```

- `Rate`. Gets or sets the actual rate for this TaxRate object. Should be entered in decimal format - i.e. .075 for 7.5%.

  ```
  public decimal Rate { set; get; }
  ```

- `TaxClassId`. Gets or sets the ID of the tax class that this rate applies to.

  ```
  public long TaxClassId { set; get; }
  ```

- `TaxRateData()`

  ```
  public TaxRateData()
  ```

- `TaxRateData(long, long, decimal)`

  ```
  public TaxRateData(long typeId, long typeItemId,
    long taxClassId, decimal rate)
  ```

- `TaxRateData(long, long, decimal, long)`

  ```
  public TaxRateData(long typeId, long typeItemId,
    long taxClassId, decimal rate, long id)
  ```

- `TypeId`. Gets or sets the type of the Tax Rate.

  ```
  public long TypeId { set; get; }
  ```

- `TypeItemId`. Gets or sets the ID of the tax type object this tax rate applies to. For instance, for a region sales tax, TypeItemId would be the ID of the corresponding region.

  ```
  public long TypeItemId { set; get; }
  ```

TaxTypeManager

9.00 and higher

The class manages commerce tax types in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.TaxTypeManager
```

Constructors

- `TaxTypeManager()`
- `TaxTypeManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 782
- *GetItem* on page 783
- *GetList* on page 785
- *Update* on page 788

Add

```
Add(Ektron.Cms.Commerce.TaxTypeData)
```

Adds a tax type to the CMS. The TaxType.Id property is populated with the new tax type's ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Tax Type Name
- * Precedence

Parameters

- `taxTypeData`. TaxTypeData object to add.

Returns

TaxTypeData object.

.aspx code snippet

```
<ol class="formFields">
<li class="clearfix">
    <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-6 last"
runat="server" Text="* Tax Type Name:" />
    <ektronUI:TextField ID="uxName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
</li>
 <div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Label ID="uxPrecedenceLabel" AssociatedControlID="uxPrecedence"
CssClass="span-6 last" runat="server" Text="* Precedence:" />
    <ektronUI:TextField ID="uxPrecedence" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
</li>
<div class="ektronTopSpace"></div>
<li class="clearfix">
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
</li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            TaxTypeManager taxTypeManager = new TaxTypeManager();
            TaxTypeData taxTypeData = new TaxTypeData();

            taxTypeData.Name = uxName.Text;
            taxTypeData.Precedence = Convert.ToInt16(uxPrecedence.Text);

            taxTypeData = taxTypeManager.Add(taxTypeData);
            MessageUtilities.UpdateMessage(uxMessage, "TaxTypeData is added with Id " +
taxTypeData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a tax type from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of tax type to delete.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxTypeIdLabel" AssociatedControlID="uxTaxTypeId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTaxTypeId" CssClass="span-6" runat="server"
```

```
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long ID = long.Parse(uxTaxTypeId.Text);

            TaxTypeManager taxTypeManager = new TaxTypeManager();
            taxTypeManager.Delete(ID);
            MessageUtilities.UpdateMessage(uxMessage, "TaxTypeData with Id " +
uxTaxTypeId.Text + " deleted", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a tax type by ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

`id`. ID of tax type to retrieve.

Returns

TaxTypeData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxTypeIdLabel" AssociatedControlID="uxTaxTypeId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTaxTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
    </ol>
    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxPrecedence" runat="server"></asp:Literal>
    </li>
 </ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Commerce;
using Ektron.Cms.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxTypeId = long.Parse(uxTaxTypeId.Text);

            TaxTypeManager taxTypeManager = new TaxTypeManager();
            TaxTypeData taxTypeData = new TaxTypeData();
            taxTypeData = taxTypeManager.GetItem(taxTypeId);
            if (taxTypeData != null)
            {

                MessageUtilities.UpdateMessage(uxMessage, "Details for Tax Type Id " +
taxTypeData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxName.Text = "Name : " + taxTypeData.Name;
                uxPrecedence.Text = "Precedence : " + taxTypeData.Precedence;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either Tax Type with ID " +
taxTypeId + " does not exist ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.TaxTypeCriteria)
```

Retrieves a list of Tax Types.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Tax Type Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter tax type being retrieved.

Returns

List of TaxTypeData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxTypePropertyLabel" AssociatedControlID="uxTaxType"
CssClass="span-5 last" runat="server" Text="* TaxType Property:" />
        <asp:DropDownList ID="uxTaxType" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
            <asp:ListItem>Precedence</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxtaxTypeDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >No records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>Id</th>
                    <th>Name</th>
                    <th>Precedence</th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
```

```
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Precedence")%>
            </td>

      </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            TaxTypeManager taxTypeManager = new TaxTypeManager();
            TaxTypeCriteria criteria = new TaxTypeCriteria();

            if (uxTaxType.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(TaxTypeProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else if (uxTaxType.SelectedItem.Text == "Name")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxTypeProperty.Name, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxTypeProperty.Precedence,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
```

```
            List<TaxTypeData> taxTypeDataList = taxTypeManager.GetList(criteria);

            uxtaxTypeDataListView.Visible = true;
            uxtaxTypeDataListView.DataSource = taxTypeDataList;
            uxtaxTypeDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.TaxTypeData)
```

Updates an existing tax type in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Tax Type Name

Parameters

- `taxTypeData`. TaxTypeData object to save.

Returns

TaxTypeData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxTypeIdLabel" AssociatedControlID="uxTaxTypeId"
CssClass="span-6 last" runat="server" Text=" Id:" />
        <ektronUI:TextField ID="uxTaxTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-6
last" runat="server" Text="* Tax Type Name:" />
```

```
        <ektronUI:TextField ID="uxName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxTypeId = long.Parse(uxTaxTypeId.Text);
            TaxTypeManager taxTypeManager = new TaxTypeManager();
            TaxTypeData taxTypeData = new TaxTypeData();
            taxTypeData = taxTypeManager.GetItem(taxTypeId);
            if (taxTypeId > 0 && taxTypeData != null)
            {
                taxTypeData.Name = uxName.Text;
                taxTypeData = taxTypeManager.Update(taxTypeData);
                MessageUtilities.UpdateMessage(uxMessage, "TaxType Updated with Id " +
uxTaxTypeId.Text, Message.DisplayModes.Success);

            }
            else
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid TaxType
ID.", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Data Classes

TaxTypeCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `TaxTypeCriteria()`

  ```
  public TaxTypeCriteria()
  ```

- `TaxTypeCriteria(Ektron.Cms.Commerce.TaxTypeProperty, EkEnumeration.OrderByDirection)`

  ```
  public TaxTypeCriteria(Ektron.Cms.Commerce.TaxTypeProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `TaxTypeProperty` are:

  - Id
  - Name
  - Precedence

TaxTypeData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `Id`. Gets or sets the name of the tax class.

  ```
  public long Id { set; get; }
  ```

- `Name`. Gets or sets the name of the tax class.

  ```
  public string Name { set; get; }
  ```

- `Precedence`. Gets or sets the precedence of the tax class. This is the priority order in which it will be applied. For example, if type "Postal Code Sales Tax is priority 1" and it exists for an address it will be used over type "State Sales Tax" which is priority 2.

  ```
  public int Precedence { set; get; }
  ```

- `TaxTypeData()`

  ```
  public TaxTypeData()
  ```

- `TaxTypeData(string, int)`

  ```
  public TaxTypeData(string name, int precedence)
  ```

- `TaxTypeData(string, int, long)`

```
public TaxTypeData(string name, int precedence, long id)
```

WarehouseManager

9.00 and higher

The class manages commerce warehouses in the CMS.

Namespace

```
Ektron.Cms.Framework.Commerce.WarehouseManager
```

Constructors

- `WarehouseManager()`
- `WarehouseManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 795
- *GetDefault* on page 797
- *GetItem* on page 799
- *GetList* on page 801
- *SetDefault* on page 804
- *Update* on page 806

Add

```
Add(Ektron.Cms.Commerce.WarehouseData)
```

Adds a warehouse to the CMS.The WarehouseData.Id property is populated with the new Warehouse ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID
- * Name
- Is Default Warehouse

Parameters

- `warehouseData`. Warehouse data object to add.

Returns

WarehouseData object added.

.aspx code snippet

```
<span>Dimensions</span>
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressLine1Label" AssociatedControlID="uxAddressLine1"
CssClass="span-3 last"
            runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxAddressLine1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCityLabel" AssociatedControlID="uxCity" CssClass="span-3
last"
            runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLagel" AssociatedControlID="uxRegionId"
CssClass="span-3 last"
            runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxPostalCodeLabel" AssociatedControlID="uxPostalCode"
CssClass="span-3 last"
            runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxPostalCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCountryIdLabel" AssociatedControlID="uxCountryId"
CssClass="span-3 last"
            runat="server" Text="* CountryId :" />
        <ektronUI:TextField ID="uxCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxWarehouseNameLabel" AssociatedControlID="uxWarehouseName"
CssClass="span-3 last" runat="server" Text="* Name:" />
        <ektronUI:TextField  ID="uxWarehouseName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxWarehouseDefaultLabel"
AssociatedControlID="uxWarehouseDefault" CssClass="span-3 last" runat="server" Text="
IsDefaultWarehouse:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxWarehouseDefault"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            bool IsDefault = false;
```

```
            WarehouseManager warehouseManager = new WarehouseManager();
            RegionManager regionManager = new RegionManager();
            CountryManager countryManager = new CountryManager();
            WarehouseData warehouseData = null;

            long regionId = long.Parse(uxRegionId.Text);
            int countryId = int.Parse(uxCountryId.Text);
            if (uxWarehouseDefault.Checked)
            {
                IsDefault = true;
            }
            RegionData regionData = regionManager.GetItem(regionId);
            CountryData countryData = countryManager.GetItem(countryId);
            AddressData addressData = new AddressData()
            {

                AddressLine1 = uxAddressLine1.Text,
                City = uxCity.Text,
                PostalCode = uxPostalCode.Text,
                Region = regionData,
                Country = countryData
            };
            warehouseData = new WarehouseData()
            {
                Address = addressData,
                IsDefaultWarehouse = IsDefault,
                Name = uxWarehouseName.Text
            };
            warehouseManager.Add(warehouseData);

            if (warehouseData.Id > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Warehouse is added with Id "
+ warehouseData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Warehouse is not added",
Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes the warehouse from CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Warehouse ID

Parameters

- `warehouseId`. ID of the warehouse to be deleted.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxWarehouseIdLabel" AssociatedControlID="uxWarehouseId"
CssClass="span-3 last" runat="server" Text="* Warehouse Id:" />
        <ektronUI:TextField  ID="uxWarehouseId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            WarehouseManager warehouseManager = new WarehouseManager();
```

```
            long warehouseId = long.Parse(uxWarehouseId.Text);
            WarehouseData WarehouseData = warehouseManager.GetItem(warehouseId);
            if (WarehouseData != null && WarehouseData.Id > 0)
            {
                warehouseManager.Delete(warehouseId);
                MessageUtilities.UpdateMessage(uxMessage, "Warehouse with Id " +
warehouseId + " is deleted.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Warehouse with Id " +
warehouseId + " does not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetDefault

```
GetDefault()
```

Gets the default warehouse.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Returns

WarehouseData object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetDefault"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
```

```
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsDefault" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAddress" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCity" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxPostalCode" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCRegionName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCountryName" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            WarehouseManager warehouseManager = new WarehouseManager();
            WarehouseData WarehouseData = warehouseManager.GetDefault();
            if (WarehouseData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for default warehouse
with ID " + WarehouseData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxName.Text  = "Name : " + WarehouseData.Name;
                uxIsDefault.Text  = "IsDefault : " + WarehouseData.IsDefaultWarehouse;
                uxAddress.Text = "Address : " + WarehouseData.Address.AddressLine1;
                uxCity.Text = "City : " + WarehouseData.Address.City;
                uxPostalCode.Text = "Postal Code : " + WarehouseData.Address.PostalCode;
                uxCRegionName.Text = "Region Name : " +
WarehouseData.Address.Region.Name;
                uxCountryName.Text = "Country Name : " +
WarehouseData.Address.Country.Name;
```

```
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Default warehouse is not
exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Gets the single warehouse data object by warehouse ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Warehouse ID

Parameters

- `warehouseId`. ID of the warehouse to be retrieved.

Returns

WarehouseData object.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxWarehouseIdLabel" AssociatedControlID="uxWarehouseId"
CssClass="span-3 last" runat="server" Text="* Warehouse Id:" />
        <ektronUI:TextField  ID="uxWarehouseId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
```

```
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsDefault" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAddress" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCity" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxPostalCode" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCRegionName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCountryName" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            WarehouseManager warehouseManager = new WarehouseManager();
            long warehouseId = long.Parse(uxWarehouseId.Text);
            WarehouseData WarehouseData = warehouseManager.GetItem(warehouseId);
            if (WarehouseData != null && WarehouseData.Id > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for Warehouse with ID
" + WarehouseData.Id.ToString() + " are below", Message.DisplayModes.Success);
```

```
                uxName.Text  = "Name : " + WarehouseData.Name;
                uxIsDefault.Text  = "IsDefault : " + WarehouseData.IsDefaultWarehouse;
                uxAddress.Text = "Address : " + WarehouseData.Address.AddressLine1;
                uxCity.Text = "City : " + WarehouseData.Address.City;
                uxPostalCode.Text = "Postal Code : " + WarehouseData.Address.PostalCode;
                uxCRegionName.Text = "Region Name : " +
WarehouseData.Address.Region.Name;
                uxCountryName.Text = "Country Name : " +
WarehouseData.Address.Country.Name;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Warehouse with Id " +
warehouseId + " does not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Commerce.WarehouseCriteria)
```

Returns the list of warehouse data object by supplied criteria.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Warehouse Property
- * Object Value

Parameters

- `criteria`. Criteria by which to filter warehouse being retrieved.

Returns

List of WarehouseCriteria filters.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
   <li class="clearfix">
        <ektronUI:Label ID="uxWarehousePropertyLabel"
AssociatedControlID="uxWarehouseProperty" CssClass="span-4 last" runat="server"
Text="Warehouse Property:" />
        <asp:DropDownList ID="uxWarehouseProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
            <asp:ListItem>AddressId </asp:ListItem>
            <asp:ListItem>City </asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxWarehouseListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                       Id
                    </th>
                    <th>
                       Name
                    </th>
                    <th>
                       Address
                    </th>
                    <th>
                        City
                    </th>
                    <th>
                       Postal Code
                    </th>
                    <th>
                       Region Name
                    </th>
                    <th>
```

```
                        Country Name
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Address.AddressLine1")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Address.City")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Address.PostalCode")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Address.Region.Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Address.Country.Name")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
```

```
            object Objectvalue;
            WarehouseCriteria criteria = new WarehouseCriteria();

            if (uxWarehouseProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(WarehouseProperty.Id,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxWarehouseProperty.SelectedItem.Text == "Name")
            {
                Objectvalue =uxObjectValue.Text;
                criteria.AddFilter(WarehouseProperty.Name ,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxWarehouseProperty.SelectedItem.Text == "AddressId")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(WarehouseProperty.AddressId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue =uxObjectValue.Text;
                criteria.AddFilter(WarehouseProperty.City ,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            WarehouseManager warehouseManger = new WarehouseManager();
            List<WarehouseData> warehouseList = warehouseManger.GetList(criteria);

            uxWarehouseListView.Visible = true;
            uxWarehouseListView.DataSource = warehouseList;
            uxWarehouseListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

SetDefault

```
SetDefault(System.Int64)
```

Marks the existing warehouse as default warehouse.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Warehouse ID

Parameters

- `warehouseId`. ID of the warehouse to become default warehouse.

.aspx code snippet

```
<ol class="formFields">
     <li class="clearfix">
        <ektronUI:Label ID="uxWarehouseIdLabel" AssociatedControlID="uxWarehouseId"
CssClass="span-3 last" runat="server" Text="* Warehouse Id:" />
        <ektronUI:TextField  ID="uxWarehouseId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
     <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="SetDefault"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            WarehouseManager warehouseManager = new WarehouseManager();
            long warehouseId = long.Parse(uxWarehouseId.Text);
            WarehouseData WarehouseData = warehouseManager.GetItem(warehouseId);
            if (WarehouseData != null && WarehouseData.Id > 0)
            {
                warehouseManager.SetDefault(warehouseId);
```

```
                MessageUtilities.UpdateMessage(uxMessage, "Warehouse with Id " +
warehouseId + " is set as default warehouse.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Warehouse with Id " +
warehouseId + " does not exists.", Message.DisplayModes.Warning);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Commerce.WarehouseData)
```

Updates the existing warehouse.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Warehouse ID
- * Address Line 1
- * City
- * Region ID
- * Postal Code
- * Country ID
- * Name
- Is Default Warehouse

Parameters

- `warehouseData`. WarehouseData object to update.

Returns

WarehouseData object updated.

.aspx code snippet

```
<span>Dimensions</span>
<ol class="formFields">

    <li class="clearfix">
        <ektronUI:Label ID="uxWarehouseIdLabel" AssociatedControlID="uxWarehouseId"
CssClass="span-3 last"
            runat="server" Text="* Warehouse Id :" />
        <ektronUI:TextField ID="uxWarehouseId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAddressLine1Label" AssociatedControlID="uxAddressLine1"
CssClass="span-3 last"
            runat="server" Text="* AddressLine1 :" />
        <ektronUI:TextField ID="uxAddressLine1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCityLabel" AssociatedControlID="uxCity" CssClass="span-3
last"
            runat="server" Text="* City :" />
        <ektronUI:TextField ID="uxCity" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRegionIdLagel" AssociatedControlID="uxRegionId"
CssClass="span-3 last"
            runat="server" Text="* RegionId :" />
        <ektronUI:TextField ID="uxRegionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPostalCodeLabel" AssociatedControlID="uxPostalCode"
CssClass="span-3 last"
            runat="server" Text="* PostalCode :" />
        <ektronUI:TextField ID="uxPostalCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxCountryIdLabel" AssociatedControlID="uxCountryId"
CssClass="span-3 last" runat="server" Text="* CountryId:" />
        <ektronUI:TextField  ID="uxCountryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxWarehouseNameLabel" AssociatedControlID="uxWarehouseName"
CssClass="span-3 last" runat="server" Text="* Name:" />
        <ektronUI:TextField  ID="uxWarehouseName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxWarehouseDefaultLabel"
AssociatedControlID="uxWarehouseDefault" CssClass="span-3 last" runat="server" Text="
```

```
IsDefaultWarehouse:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxWarehouseDefault"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Commerce;
using Ektron.Cms.Framework.Commerce;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            bool IsDefault = false;
            WarehouseManager warehouseManager = new WarehouseManager();
            RegionManager regionManager = new RegionManager();
            CountryManager countryManager = new CountryManager();

            long regionId = long.Parse(uxRegionId.Text);
            int countryId = int.Parse(uxCountryId.Text);
            long warehouseId=long.Parse(uxWarehouseId.Text);
            if (uxWarehouseDefault.Checked)
            {
                IsDefault = true;
            }
            RegionData regionData = regionManager.GetItem(regionId);
            CountryData countryData = countryManager.GetItem(countryId);
            WarehouseData warehouseData = warehouseManager.GetItem(warehouseId);
            if (warehouseData != null)
            {
                AddressData addressData = warehouseData.Address ;
                addressData.AddressLine1 =uxAddressLine1.Text ;
                addressData.City =uxCity.Text ;
                addressData.PostalCode =uxPostalCode .Text ;
                addressData.Region =regionData ;
                addressData.Country =countryData ;
```

```
                warehouseData.Address=addressData ;
                warehouseData.Name =uxWarehouseName.Text ;
                warehouseData.IsDefaultWarehouse =IsDefault ;

                warehouseManager.Update(warehouseData);
                MessageUtilities.UpdateMessage(uxMessage, "Warehouse is updated with Id
" + warehouseData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Warehouse with Id "+
warehouseId + " does not exists.", Message.DisplayModes.Warning );
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

WarehouseCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Commerce
```

Constructors

- `WarehouseCriteria()`

  ```
  public WarehouseCriteria()
  ```

- `WarehouseCriteria(Ektron.Cms.Commerce.WarehouseProperty, EkEnumeration.OrderByDirection)`

  ```
  public WarehouseCriteria(Ektron.Cms.Commerce.WarehouseProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `WarehouseProperty` are:

  - AddressCompanyName
  - AddressId
  - AddressLine1
  - AddressLine2
  - City
  - CounteryEnabled
  - CountryId

- CountryLongIsoCode
- CountryName
- CountryShortIsoCode
- Id
- IsCommercial
- IsDefault
- IsValidated
- Name
  phone
- PostalCode
- RegionAlphaCode
- RegionEnabled
- RegionId
- RegionName

WarehouseData

Namespace

```
Ektron.Cms.Commerce
```

Properties

- `Address`. Gets or sets the address of this Warehouse.

  ```
  public Ektron.Cms.Commerce.AddressData Address { set; get; }
  ```

- `Id`. Gets or sets Id of Address.

  ```
  public long Id { set; get; }
  ```

- `IsDefaultWarehouse`. Gets or sets the IsDefaultWarehouse status property for the warehouse. If true, this warehouse is the default warehouse for the system.

  ```
  public bool IsDefaultWarehouse { set; get; }
  ```

- `Name`. Gets or sets the name of this Warehouse.

  ```
  public string Name { set; get; }
  ```

- `WarehouseData()`

  ```
  public WarehouseData()
  ```

- `WarehouseData(string, Ektron.Cms.Commerce.AddressData)`

  ```
  public WarehouseData(string warehouseName,
    Ektron.Cms.Commerce.AddressData address)
  ```

Community

8.50 and higher

The Community manager category manages social media aspects of a website and has the following classes:

- *CommunityGroupManager* on the next page. Manages groups of users.
- *FavoriteManager* on page 867. Manages a list of Ektron CMS content and external Web links (URLs) that a user has designated as favorite content.
- *FavoriteTaxonomyManager* on page 882. Manages user favorite taxonomies.
- *FlagManager* on page 896. Manages flags that are assigned to content to provide feedback.
- *FriendsManager* on page 910. Manages colleague connections (friends).
- *FriendsTaxonomyManager* on page 939. Manages taxonomies of colleague connections (friends).
- *MessageBoardManager* on page 953. Manages items that are posted on a message board.
- *MicromessageManager* on page 979. Manages user status messages that are posted to an activity stream.
- *PrivateMessageManager* on page 1005. Manages messages between users.
- *RatingManager* on page 1017. 8.60 and higher Manages user ratings of content.
- *TagManager* on page 1042. Manages tags.

CommunityGroupManager

8.50 and higher

The CommunityGroupManager class manages groups of users.

Namespace

```
Ektron.Cms.Framework.Community.CommunityGroupManager
```

Constructors

- `CommunityGroupManager()`
- `CommunityGroupManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `CommunityGroupManagerService`. Returns an instance of the business objects task manager service.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *GetCalendarId* on page 831
- *GetDiscussionBoardId* on page 833
- *GetFolderId* on page 834
- *GetItem* on page 836
- *GetList (CommunityGroupCriteria)* on page 838 (CommunityGroupCriteria)
- *GetList (UserToCommunityGroupCriteria)* on page 841 (UserToCommunityGroupCriteria)
- *GetMemberStatus* on page 843
- *GetTaxonomyId* on page 845
- *GetUserList* on page 847
- *Invite* on page 849 (email)
- *Invite* on page 850 (user)
- *IsUserInGroup* on page 852
- *RemoveUser* on page 853
- *RequestJoin* on page 855
- *Update* on page 856

AcceptInvite

```
AcceptInvite(System.Int64,System.Int64)
```

Accepts an invitation of a community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID
- * User ID

Parameters

- `communityGroupId`. ID of the community group.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-6 last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="8" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Accept Invite"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            long userId = long.Parse(uxUserId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            communityGroupManager.AcceptInvite(communityGroupId, userId);

            MessageUtilities.UpdateMessage(uxMessage, "Invite accepted by User with Id "
+ userId, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Add

```
Add(Ektron.Cms.CommunityGroupData)
```

Adds a community group.

When you add the community group data, the method returns a custom CMSData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Group Name
- Enable Distribute
- Enable Group Mail
- Group Image
- * Group Admin Id
- Short Description
- Long Description

Parameters

- `communityGroupData`. The CommunityGroupData object to add.

Returns

Returns the custom CommunityGroupData object added.

.aspx code snippet

```
<ol class="formFields">
<li class="clearfix">
    <ektronUI:Label ID="uxGroupNameLabel" AssociatedControlID="uxGroupName"
CssClass="span-4 last" runat="server" Text="* Group Name:" />
    <ektronUI:TextField  ID="uxGroupName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxGroupEnableDistributeLabel"
AssociatedControlID="uxGroupEnableDistribute" CssClass="span-4 last" runat="server"
Text="Group Enable Distribute:" />
    <asp:CheckBox id="uxGroupEnableDistribute" runat="server" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxEnableGroupEmailsLabel"
AssociatedControlID="uxEnableGroupEmails" CssClass="span-4 last" runat="server"
Text="Enable Group Email:" />
    <asp:checkbox ID="uxEnableGroupEmails" runat="server" AutoPostBack="true"
OnCheckedChanged="uxEnableGroupEmails_OnCheckedChanged" />
</li>
<li class="clearfix">
```

```
    <ektronUI:Label ID="uxEmailAccountNameLabel"
AssociatedControlID="uxEmailAccountName" CssClass="span-4 last" runat="server"
Text="Email Account Name:" Enabled="false" />
    <ektronUI:TextField  ID="uxEmailAccountName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Enabled="false" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxEmailAddressLabel" AssociatedControlID="uxEmailAddress"
CssClass="span-4 last" runat="server" Text="Email Address:" Enabled="false" />
    <ektronUI:TextField  ID="uxEmailAddress" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Enabled="false" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxEmailPasswordLabel" AssociatedControlID="uxEmailPassword"
CssClass="span-4 last" runat="server" Text="Email Password:" Enabled="false" />
    <ektronUI:TextField  ID="uxEmailPassword" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Enabled="false" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxGroupImageLabel" AssociatedControlID="uxGroupImage"
CssClass="span-4 last" runat="server" Text="Group Image:" />
    <asp:FileUpload ID="uxGroupImage" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" size="29.75" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxGroupAdminLabel" AssociatedControlID="uxGroupAdmin"
CssClass="span-4 last" runat="server" Text="* Group Admin Id:" />
    <ektronUI:TextField  ID="uxGroupAdmin" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxGroupShortDescriptionLabel"
AssociatedControlID="uxGroupShortDescription" CssClass="span-4 last" runat="server"
Text="Short Description:" />
    <ektronUI:TextField  ID="uxGroupShortDescription" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxGroupLongDescriptionLabel"
AssociatedControlID="uxGroupLongDescription" CssClass="span-4 last" runat="server"
Text="Long Description:" />
    <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxGroupLongDescription"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4 last" runat="server" Text="* -
Required" />
    <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
</li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Storage;
using Microsoft.VisualBasic;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager Usermanager = new UserManager();
            long groupAdminId;
            long.TryParse(uxGroupAdmin.Text, out groupAdminId);
            if (!String.IsNullOrWhiteSpace(uxGroupName.Text))
            {
                UserData userBaseData = Usermanager.GetItem(groupAdminId);
                List<UserData> userlist = new List<UserData>();
                if (userBaseData != null && userBaseData.Id != 0)
                {
                    userlist.Add(userBaseData);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Group
Admin Id.", Message.DisplayModes.Error);
                    uxPageMultiView.SetActiveView(uxViewMessage);
                    return;
                }

                CommunityGroupManager communityGroupManager = new CommunityGroupManager
();
                CommunityGroupData communityGroupData = new CommunityGroupData()
                {
                    Name = uxGroupName.Text,
                    EnableDistributeToSite = uxGroupEnableDistribute.Checked,
                    LongDescription = uxGroupLongDescription.Text,
                    EnableGroupEmail = uxEnableGroupEmails.Checked,
                    Admins = userlist,
                };

                if (communityGroupData.EnableGroupEmail == true)
                {
                    if (!String.IsNullOrWhiteSpace(uxEmailAccountName.Text) &&
!String.IsNullOrWhiteSpace(uxEmailAddress.Text) && !String.IsNullOrWhiteSpace
(uxEmailPassword.Text))
                    {
```

```
                        String expression = @"^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-9]
{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";
                        if (!System.Text.RegularExpressions.Regex.IsMatch
(uxEmailAddress.Text, expression))
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid EmailAddress.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
                            return;
                        }
                        communityGroupData.EmailAccountName = uxEmailAccountName.Text;
                        communityGroupData.EmailAddress = uxEmailAddress.Text;
                        communityGroupData.EmailPassword = uxEmailPassword.Text;
                    }
                    else
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "Need valid
EmailAddress, EmailPassword, and EmailAccountName when EnableGroupEmail is enabled.",
Message.DisplayModes.Error);
                        uxPageMultiView.SetActiveView(uxViewMessage);
                        return;
                    }
                }

                if (uxGroupImage.HasFile)
                {
                    if ((uxGroupImage.PostedFile.ContentType == "image/pjpeg") ||
(uxGroupImage.PostedFile.ContentType == "image/jpeg") ||
(uxGroupImage.PostedFile.ContentType == "image/gif"))
                    {
                        string[] fileNameArray = Strings.Split
(uxGroupImage.PostedFile.FileName, "\\", -1, 0);
                        string strFileName = (System.Guid.NewGuid().ToString
()).Substring(0, 5) + "_g_" + fileNameArray[(fileNameArray.Length - 1)];
                        StorageClient.Context.File.UploadStream
(uxGroupImage.PostedFile.InputStream, Server.MapPath(communityGroupManager.SitePath +
"uploadedimages/" + strFileName));
                        Utilities.ProcessThumbnail(Server.MapPath
(communityGroupManager.SitePath + "uploadedimages/"), strFileName);
                        communityGroupData.Image = communityGroupManager.SitePath +
"uploadedimages/thumb_" + Utilities.GetCorrectThumbnailFileWithExtn(strFileName);
                    }
                    else
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "Not a valid jpeg/gif
image.", Message.DisplayModes.Error);
                        uxPageMultiView.SetActiveView(uxViewMessage);
                        return;
                    }
                }

                if (!string.IsNullOrWhiteSpace(uxGroupShortDescription.Text))
                {
                    if (uxGroupShortDescription.Text.Length > 100)
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "Short Description
```

```
should not exceed 100 characters.", Message.DisplayModes.Error);
                        uxPageMultiView.SetActiveView(uxViewMessage);
                        return;
                    }
                    communityGroupData.ShortDescription = uxGroupShortDescription.Text;
                }

                if (communityGroupManager.RequestInformation.UserId > 0)
                {
                    communityGroupManager.Add(communityGroupData);
                    MessageUtilities.UpdateMessage(uxMessage, "Community Group Added
with Id " + communityGroupData.Id, Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Login required to add a
community group", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Group
Name.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

AddUser

```
AddUser(System.Int64,System.Int64)
```

Adds a user to a community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID
- * User ID

Parameters

- `communityGroupId`. ID of the community group.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="8" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Add
User"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            long userId = long.Parse(uxUserId.Text);

            UserManager Usermanager = new UserManager();
            UserData Userdata = Usermanager.GetItem(userId);

            if (Userdata != null)
            {
                CommunityGroupManager communityGroupManager = new CommunityGroupManager
```

```
();
                communityGroupManager.AddUser(communityGroupId, userId);

                MessageUtilities.UpdateMessage(uxMessage,  "User with ID " + userId + "
is added to Community Group with ID " + communityGroupId, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "User with ID " + userId + "
does not exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

ApproveJoin

```
DeclineJoin(System.Int64,System.Int64)
```

Accepts a join request of a community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID
- * User ID

Parameters

- `communityGroupId`. ID of the community group.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-6 last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="8" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="ApproveJoin"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            long userId = long.Parse(uxUserId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            communityGroupManager.ApproveJoin(communityGroupId, userId);

            MessageUtilities.UpdateMessage(uxMessage, "Join accepted by User with Id " +
userId, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeclineInvite

```
DeclineInvite(System.Int64,System.Int64)
```

Declines an invitation of a community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID
- * User ID

Parameters

- `communityGroupId`. ID of the community group.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="8" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Decline Invite"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            long userId = long.Parse(uxUserId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            communityGroupManager.DeclineInvite(communityGroupId, userId);

            MessageUtilities.UpdateMessage(uxMessage, "Invite declined by User with Id "
+ userId, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeclineJoin

```
DeclineJoin(System.Int64,System.Int64)
```

Declines a join request of a community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID
- * User ID

Parameters

- `communityGroupId`. ID of the community group.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
```

```
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="8" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Decline Join"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            long userId = long.Parse(uxUserId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            communityGroupManager.DeclineJoin(communityGroupId, userId);

            MessageUtilities.UpdateMessage(uxMessage, "Join declined by User with Id " +
userId, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a community group from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `ID`. ID of the community group to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-3 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix" style="color: red;">
                    <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                        Text="Status:" />
                </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms.Framework.Community;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            CommunityGroupData communityGroupData = communityGroupManager.GetItem
(communityGroupId);

            if (communityGroupData != null)
            {
                communityGroupManager.Delete(communityGroupId);
                MessageUtilities.UpdateMessage(uxMessage, "Community Group with Id " +
communityGroupId + " deleted.", Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
CommunityGroup Id.", Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetAdminList

```
GetAdminList(System.Int64)
```

Retrieves a list of administrator users for a community group ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID

Parameters

- `communityGroupId`. ID of the community group.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-3 last" runat="server" Text="*
Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Message ID="uxStatus" DisplayMode="Error" runat="server"
Visible="false" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetAdminList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
<asp:ListView ID="uxUserlist" runat="server" ItemPlaceholderID="aspItemPlaceholder"
Visible="false">
<EmptyDataTemplate>
    Criteria did not match any records.</EmptyDataTemplate>
<LayoutTemplate>
    <table class="devsite-api-method">
        <thead>
            <tr>
                <th>
                    Id
                </th>
                <th>
                    FirstName
                </th>
                <th>
                    LastName
                </th>
                <th>
                    DisplayName
                </th>
            </tr>
        </thead>
        <tbody>
            <asp:PlaceHolder ID="aspItemPlaceholder" runat="server"></asp:PlaceHolder>
        </tbody>
    </table>
```

```
</LayoutTemplate>
<ItemTemplate>
    <tr>
        <td class="devsite-method">
            <%# Eval("Id")%>
        </td>
        <td class="devsite-method">
            <%# Eval("FirstName")%>
        </td>
        <td class="devsite-method">
            <%# Eval("LastName")%>
        </td>
        <td class="devsite-method">
            <%# Eval("DisplayName")%>
        </td>
    </tr>
</ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            CommunityGroupData communityGroupData = communityGroupManager.GetItem
(communityGroupId);

            if (communityGroupData != null)
            {
                List<UserData> userlist = communityGroupManager.GetAdminList
(communityGroupId);

                uxUserlist.Visible = true;
                uxUserlist.DataSource = userlist;
                uxUserlist.DataBind();
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
CommunityGroup Id.", Message.DisplayModes.Error);
```

```
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetBlogId

```
GetBlogId(System.Int64)
```

Retrieves the blog ID of the community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID

Parameters

- `communityGroupId`. ID of the community group.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Blog Id"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            long blogId = communityGroupManager.GetBlogId(communityGroupId);

            if (blogId == 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either the community group
does not exists or the community group has no blog.", Message.DisplayModes.Error);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Blog Id : " + blogId,
Message.DisplayModes.Success);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetCalendarId

```
GetCalendarId(System.Int64)
```

Retrieves the calendar ID of the community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID

Parameters

- `communityGroupId`. ID of the community group.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Calendar Id"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            long calendarId = communityGroupManager.GetCalendarId(communityGroupId);

            if (calendarId == 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either the Community group
does not exists or or the community group has no calendar.",
Message.DisplayModes.Error);
            }
```

```
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Calendar Id  : " +
calendarId, Message.DisplayModes.Success);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetDiscussionBoardId

```
GetDiscussionBoardId(System.Int64)
```

Retrieves the discussion board ID of the community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID

Parameters

- `communityGroupId`. ID of the community group.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Discussion Board Id"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            long discussionBoardId = communityGroupManager.GetDiscussionBoardId
(communityGroupId);

            if (discussionBoardId == 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Either the Community group
does not exists or the community group has no discussion board.",
Message.DisplayModes.Error);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Discussion Board Id : " +
discussionBoardId, Message.DisplayModes.Success);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetFolderId

```
GetFolderId(System.Int64)
```

Retrieves the folder ID of the community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID

Parameters

- `communityGroupId`. ID of the community group.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Folder Id"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            long folderId = communityGroupManager.GetFolderId(communityGroupId);

            if (folderId == 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Community group does not
```

```
exists.", Message.DisplayModes.Error);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Folder Id : " + folderId,
Message.DisplayModes.Success);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific CommunityGroupData object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID

Parameters

- `id`. ID of the community group to get.

Returns

Community group details in a CommunityGroupData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-3 last" runat="server" Text="*
Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxGroupName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxGroupAdmin" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxGroupEnableDistribute" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxGroupLongDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxGroupShortDescription" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            CommunityGroupData communityGroupData = communityGroupManager.GetItem
(communityGroupId);

            if(communityGroupData != null)
            {
               MessageUtilities.UpdateMessage(uxMessage, "Details for community group
with ID " + communityGroupData.Id.ToString() + " are below",
Message.DisplayModes.Success);
```

```
                uxGroupName.Text = "Group Name : " + communityGroupData.Name;
                uxGroupEnableDistribute.Text = "Group Enable Distribute : " +
communityGroupData.EnableDistributeToSite;
                uxGroupLongDescription.Text = "Group Long Description : " +
communityGroupData.LongDescription;
                uxGroupShortDescription.Text = "Group Short Description : " +
communityGroupData.ShortDescription;
                uxGroupAdmin.Text = "Group Admin :" + communityGroupData.Admins
[0].Username;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Community group does not
exist. ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (CommunityGroupCriteria)

```
GetList(Ektron.Cms.Community.CommunityGroupCriteria)
```

Retrieves community groups associated with the specified users. Username or UserId are required criteria for this method to return data.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Community Group Property
- * Object Value

Parameters

- `criteria`. Used to specify, or filter, the community groups to return.

Returns

A list of community groups.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupPropertyLabel"
AssociatedControlID="uxCommunityGroupProperty" CssClass="span-4 last" runat="server"
Text="CommunityGroupProperty:" />
        <asp:DropDownList ID="uxCommunityGroupProperty" runat="server">
            <asp:ListItem>Group Id</asp:ListItem>
            <asp:ListItem>Group Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxcommunityGroupListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Group Name
                    </th>
                    <th>
                        Group Id
                    </th>
                    <th>
                        Group Long Description
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("GroupName")%>
```

```
            </td>
            <td class="devsite-method">
                <%# Eval("GroupId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("GroupLongDescription")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
                        object Objectvalue;

            Ektron.Cms.Community.CommunityGroupCriteria criteria = new
Ektron.Cms.Community.CommunityGroupCriteria();
            if (uxCommunityGroupProperty.SelectedItem.Text == "Group Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(CommunityGroupProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CommunityGroupProperty.Name,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            List<CommunityGroupData> communityGroupList = communityGroupManager.GetList
(criteria);

            uxcommunityGroupListView.Visible = true;
            uxcommunityGroupListView.DataSource = communityGroupList;
            uxcommunityGroupListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (UserToCommunityGroupCriteria)

8.60 and higher

```
GetList(Ektron.Cms.Community.UserToCommunityGroupCriteria)
```

Returns a list of CommunityGroups that fit the supplied criteria, which in this case can be all the groups given a supplied user id.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- User To Community Property
- * Object Value

Parameters

- `criteria`. Used to specify, or filter, the community groups to return.

Returns

A list of community groups.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserToCommunityGroupPropertyLabel"
AssociatedControlID="uxUserToCommunityGroupProperty" CssClass="span-4 last"
runat="server" Text="UserToCommunityProperty:" />
        <asp:DropDownList ID="uxUserToCommunityGroupProperty" runat="server">
            <asp:ListItem>User Id</asp:ListItem>
            <asp:ListItem>User Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
```

```
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxcommunityGroupListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Group Name
                    </th>
                    <th>
                        Group Id
                    </th>
                    <th>
                        Group Long Description
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("GroupName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("GroupId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("GroupLongDescription")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Community;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
 protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
                        object Objectvalue;
            UserToCommunityGroupCriteria criteria = new UserToCommunityGroupCriteria();

            criteria.OrderByField = UserToCommunityProperty.Id ;
            criteria.OrderByDirection = EkEnumeration.OrderByDirection.Ascending;

            if (uxUserToCommunityGroupProperty.SelectedItem.Text == "User Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(UserToCommunityProperty.UserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(UserToCommunityProperty.UserName ,
CriteriaFilterOperator.EqualTo , Objectvalue);
            }

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            List<CommunityGroupData> communityGroupList = communityGroupManager.GetList
(criteria);

            uxcommunityGroupListView.Visible = true;
            uxcommunityGroupListView.DataSource = communityGroupList;
            uxcommunityGroupListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetMemberStatus

```
GetMemberStatus(System.Int64,System.Int64)
```

Retrieves the member status of the user in the community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID
- * User ID

Parameters

- `communityGroupId`. ID of the community group.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Member Status"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            long userId = long.Parse(uxUserId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            EkEnumeration.GroupMemberStatus groupMemberStatus =
communityGroupManager.GetMemberStatus(communityGroupId, userId);

            MessageUtilities.UpdateMessage(uxMessage, "Group Member Status : " +
groupMemberStatus, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTaxonomyId

```
GetTaxonomyId(System.Int64)
```

Retrieves the taxonomy ID of the community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID

Parameters

- `communityGroupId`. ID of the community group.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
```

```
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Taxonomy Id"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            long taxonomyId = communityGroupManager.GetTaxonomyId(communityGroupId);

            if (taxonomyId == 0)
            {
                MessageUtilities.UpdateMessage(uxMessage,"Either the Community group
does not exists or the community group has no taxonomy.", Message.DisplayModes.Error);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy Id : " + taxonomyId,
Message.DisplayModes.Success);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

GetUserList

```
GetUserList(System.Int64)
```

Retrieves a list of directory users that fit the community group ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `communityGroupId`. ID of the community group.

Returns

A list of directory users.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-3 last" runat="server" Text="*
Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
User List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxcommunityUserListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
```

```
                        User Name
                    </th>
                    <th>
                        User Id
                    </th>
                    <th>
                        Email
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Username")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Email")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            string userEmail = uxUserEmail.Text;

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            long inviteId = communityGroupManager.Invite(communityGroupId, userEmail);

            MessageUtilities.UpdateMessage(uxMessage, "Invite sent to User",
Message.DisplayModes.Success);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Invite

```
Invite(System.Int64,System.String)
```

Invites a email to a community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID
- * User ID

Parameters

- `communityGroupId`. ID of the community group.
- `email`. The email address of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="8" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Invite"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
```

```
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            long userId = long.Parse(uxUserId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            long inviteId = communityGroupManager.Invite(communityGroupId, userId);

            MessageUtilities.UpdateMessage(uxMessage, "Invite sent to User",
Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Invite

```
Invite(System.Int64,System.Int64)
```

Invites a user to a community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID
- * User ID

Parameters

- `communityGroupId`. ID of the community group.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="8" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Invite"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            long userId = long.Parse(uxUserId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            long inviteId = communityGroupManager.Invite(communityGroupId, userId);
```

```
            MessageUtilities.UpdateMessage(uxMessage, "Invite sent to User",
Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IsUserInGroup

```
IsUserInGroup(System.Int64,System.Int64)
```

Determines if a user is in a group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID
- * User ID

Parameters

- `communityGroupId`. ID of the community group.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-6 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-6 last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Is
User In Group"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            long userId = long.Parse(uxUserId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            Boolean isUserInGroup = communityGroupManager.IsUserInGroup(userId,
communityGroupId);

            MessageUtilities.UpdateMessage(uxMessage, "Is User with ID " + userId + "
present in in Community Group with ID " + communityGroupId + " : " + isUserInGroup,
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

RemoveUser

```
RemoveUser(System.Int64,System.Int64)
```

Removes a user from a community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID
- * User ID

Parameters

- `communityGroupId`. ID of the community group.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Community Group Id :" />
        <ektronUI:TextField ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="10005" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Remove User"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
```

```
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            long userId = long.Parse(uxUserId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            communityGroupManager.RemoveUser(communityGroupId, userId);

            MessageUtilities.UpdateMessage(uxMessage, "User with ID " + userId + "
removed from Community Group with ID " + communityGroupId,
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

RequestJoin

```
RequestJoin(System.Int64,System.Int64)
```

Creates a request to join a community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Group ID
- * User ID

Parameters

- `communityGroupId`. ID of the community group.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="Label1" AssociatedControlID="uxCommunityGroupId"
CssClass="span-4 last" runat="server" Text="* Community Group Id :" />
        <ektronUI:TextField ID="TextField1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="Label2" AssociatedControlID="uxUserId" CssClass="span-4
```

```
last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="TextField2" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="8" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="Button1" runat="server" OnClick="uxSubmit_Click"
Text="Request Join"></ektronUI:Button>
        <ektronUI:Label ID="Label3" CssClass="span-4" runat="server" Text="* - Required"
/>
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId = long.Parse(uxCommunityGroupId.Text);
            long userId = long.Parse(uxUserId.Text);

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            communityGroupManager.RequestJoin(communityGroupId, userId);

            MessageUtilities.UpdateMessage(uxMessage, "User with Id " + userId + "
requested to join ", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.CommunityGroupData)
```

Updates an existing community group item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Group ID
- Group Name
- Enable Distribute
- Enable Group Mail
- Short Description
- Long Description

Parameters

- `communityGroupData`. The CommunityGroupData object to update.

Returns

Returns the custom CommunityGroupData object that was updated.

Remarks

Validate the community group item with `GetItem()` before you update the properties. Then, call `Update` with your modified CommunityGroupData object.

NOTE: You cannot change all properties after the initial Add event, such as the CommunityGroupData.Type.

.aspx code snippet

```
<ol class="formFields">
<li class="clearfix">
    <ektronUI:Label ID="uxCommunityGroupIdLabel"
AssociatedControlID="uxCommunityGroupId" CssClass="span-4 last" runat="server" Text="*
Group Id:" />
    <ektronUI:TextField  ID="uxCommunityGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxGroupNameLabel" AssociatedControlID="uxGroupName"
CssClass="span-4 last" runat="server" Text=" Group Name:" />
    <ektronUI:TextField  ID="uxGroupName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxGroupEnableDistributeLabel"
AssociatedControlID="uxGroupEnableDistribute" CssClass="span-4 last" runat="server"
Text="Group Enable Distribute:" />
    <asp:checkbox ID="uxGroupEnableDistribute" runat="server" />
```

```
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxEnableGroupEmailsLabel"
AssociatedControlID="uxEnableGroupEmails" CssClass="span-4 last" runat="server" Text="
Enable Group Email:" />
    <asp:checkbox ID="uxEnableGroupEmails"  runat="server" AutoPostBack="true"
OnCheckedChanged="uxEnableGroupEmails_OnCheckedChanged"  />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxEmailAccountNameLabel"
AssociatedControlID="uxEmailAccountName" CssClass="span-4 last" runat="server"
Text="Email Account Name:" Enabled="false" />
    <ektronUI:TextField  ID="uxEmailAccountName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Enabled="false" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxEmailAddressLabel" AssociatedControlID="uxEmailAddress"
CssClass="span-4 last" runat="server" Text="Email Address:" Enabled="false" />
    <ektronUI:TextField  ID="uxEmailAddress" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Enabled="false" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxEmailPasswordLabel" AssociatedControlID="uxEmailPassword"
CssClass="span-4 last" runat="server" Text="Email Password:" Enabled="false" />
    <ektronUI:TextField  ID="uxEmailPassword" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Enabled="false" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxGroupImageLabel" AssociatedControlID="uxGroupImage"
CssClass="span-4 last" runat="server" Text="Group Image:" />
    <asp:FileUpload ID="uxGroupImage" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" size="29.75" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxGroupAdminLabel" AssociatedControlID="uxGroupAdmin"
CssClass="span-4 last" runat="server" Text="Group Admin Id:" />
    <ektronUI:TextField  ID="uxGroupAdmin" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxGroupShortDescriptionLabel"
AssociatedControlID="uxGroupShortDescription" CssClass="span-4 last" runat="server"
Text="Short Description:" />
    <ektronUI:TextField  ID="uxGroupShortDescription" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxGroupLongDescriptionLabel"
AssociatedControlID="uxGroupLongDescription" CssClass="span-4 last" runat="server"
Text="Long Description:" />
    <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxGroupLongDescription"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
</li>
<li class="clearfix">
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4 last" runat="server" Text="* -
Required" />
```

```
    <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
</li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Storage;
using Microsoft.VisualBasic;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityGroupId;
            long.TryParse(uxCommunityGroupId.Text, out communityGroupId);
            if (communityGroupId != 0)
            {
                CommunityGroupManager communityGroupManager = new CommunityGroupManager
();
                CommunityGroupData communityGroupData = communityGroupManager.GetItem
(communityGroupId);

                if (communityGroupData != null)
                {
                    communityGroupData.Name = uxGroupName.Text != string.Empty ?
uxGroupName.Text : communityGroupData.Name;
                    communityGroupData.EnableDistributeToSite =
uxGroupEnableDistribute.Checked == true ? uxGroupEnableDistribute.Checked :
communityGroupData.EnableDistributeToSite;
                    communityGroupData.LongDescription = uxGroupLongDescription.Text !=
string.Empty ? uxGroupLongDescription.Text : communityGroupData.LongDescription;

                    if (uxEnableGroupEmails.Checked == true)
                    {
                        if (communityGroupData.EnableGroupEmail == true)
                        {
                            communityGroupData.EmailAccountName =
uxEmailAccountName.Text != "" ? uxEmailAccountName.Text :
communityGroupData.EmailAccountName;
                            if (uxEmailAddress.Text != "")
                            {
                                EmailAddressValidation(uxEmailAddress.Text);
```

```
                                communityGroupData.EmailAddress = uxEmailAddress.Text;
                            }
                            communityGroupData.EmailPassword = uxEmailPassword.Text !=
"" ? uxEmailPassword.Text : communityGroupData.EmailPassword;
                        }
                        else
                        {
                            if (!String.IsNullOrWhiteSpace(uxEmailAccountName.Text) &&
!String.IsNullOrWhiteSpace(uxEmailAddress.Text) && !String.IsNullOrWhiteSpace
(uxEmailPassword.Text))
                            {
                                EmailAddressValidation(uxEmailAddress.Text);
                                communityGroupData.EmailAccountName =
uxEmailAccountName.Text;
                                communityGroupData.EmailAddress = uxEmailAddress.Text;
                                communityGroupData.EmailPassword = uxEmailPassword.Text;
                            }
                            else
                            {
                                MessageUtilities.UpdateMessage(uxMessage, "Need valid
EmailAddress, EmailPassword, and EmailAccountName when EnableGroupEmail is enabled.",
Message.DisplayModes.Error);
                                uxPageMultiView.SetActiveView(uxViewMessage);
                                return;
                            }
                        }
                        communityGroupData.EnableGroupEmail =
uxEnableGroupEmails.Checked;
                    }

                    if (uxGroupImage.HasFile)
                    {
                        if ((uxGroupImage.PostedFile.ContentType == "image/pjpeg") ||
(uxGroupImage.PostedFile.ContentType == "image/jpeg") ||
(uxGroupImage.PostedFile.ContentType == "image/gif"))
                        {
                            string[] fileNameArray = Strings.Split
(uxGroupImage.PostedFile.FileName, "\\", -1, 0);
                            string strFileName = (System.Guid.NewGuid().ToString
()).Substring(0, 5) + "_g_" + fileNameArray[(fileNameArray.Length - 1)];
                            StorageClient.Context.File.UploadStream
(uxGroupImage.PostedFile.InputStream, Server.MapPath(communityGroupManager.SitePath +
"uploadedimages/" + strFileName));
                            Utilities.ProcessThumbnail(Server.MapPath
(communityGroupManager.SitePath + "uploadedimages/"), strFileName);
                            communityGroupData.Image = communityGroupManager.SitePath +
"uploadedimages/thumb_" + Utilities.GetCorrectThumbnailFileWithExtn(strFileName);
                        }
                        else
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "Not a valid
jpeg/gif image.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
                            return;
                        }
                    }
```

```
                    if (!string.IsNullOrWhiteSpace(uxGroupAdmin.Text))
                    {
                        long groupAdminId;
                        long.TryParse(uxGroupAdmin.Text, out groupAdminId);
                        UserManager Usermanager = new UserManager();
                        UserData userBaseData = Usermanager.GetItem(groupAdminId);
                        List<UserData> userList = new List<UserData>();

                        if (userBaseData != null && userBaseData.Id != 0)
                        {
                            userList.Add(userBaseData);
                        }
                        else
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid Group Admin Id.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
                            return;
                        }
                        communityGroupData.Admins = userList;
                    }

                    if (!string.IsNullOrWhiteSpace(uxGroupShortDescription.Text))
                    {
                        if (uxGroupShortDescription.Text.Length > 100)
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "Short Description
should not exceed 100 characters.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
                            return;
                        }
                        communityGroupData.ShortDescription =
uxGroupShortDescription.Text;
                    }

                    communityGroupManager.Update(communityGroupData);

                    MessageUtilities.UpdateMessage(uxMessage, "Community group with Id "
+ communityGroupData.Id + " updated.", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Community group does not
exist for given Group Id.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid Community
Group Id.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
```

```
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

CommunityGroupCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Community
```

Constructors

- `CommunityGroupCriteria()`

  ```
  public CommunityGroupCriteria()
  ```

- `CommunityGroupCriteria(Ektron.Cms.CommunityGroupProperty, EkEnumeration.OrderByDirection)`

  ```
  public CommunityGroupCriteria(Ektron.Cms.CommunityGroupProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `CommunityGroupProperty` are:

  - AdminId
  - CreatedUser
  - DateCreated
  - DateModified
  - EnableCalendar
  - EnableDiscussionBoard
  - EnableDistribute
  - EnableDocumentsInNotifications
  - EnableGroupEmail
  - EnableMembersManageFolders
  - EnableTodoList
  - Enroll
  - FolderId
  - GroupEnroll
  - GroupId
  - GroupImage
  - GroupLevel
  - GroupLocation
  - GroupName

- GroupParentId
- GroupPath
- Id
- Image
- Level
- Location
- LongDescription
- Name
- ParentId
- Path
- ShortDescription

Properties

- `ReturnAdminList`

```
public bool ReturnAdminList { set; get; }
```

CommunityGroupData

Namespace

```
Ektron.Cms
```

Properties

- `Admin`

```
public Ektron.Cms.UserBaseData Admin { set; get; }
```

- `Admins`

```
public System.Collections.Generic.List<UserData> Admins { set; get; }
```

- `AllowMemberstoManageFolders`

```
public bool AllowMembersToManageFolders { set; get; }
```

- `Category`

```
public string Category { set; get; }
```

- `CreatedDate`

```
public System.DateTime CreatedDate { set; get; }
```

- `CreatedUser`

```
public long CreatedUser { set; get; }
```

- `EmailAccountName`. Email Account name assigned to community group. This is required for checking group email.

```
public string EmailAccountName { set; get; }
```

- `EmailAddress`. Email address assigned to Community Group.

```
public string EmailAddress { set; get; }
```

- `EmailPassword`. Password for email account assigned to community group. This is required for checking group email.

```
public string EmailPassword { set; get; }
```

- `EnableCalendar`

```
public bool EnableCalendar { set; get; }
```

- `EnableDiscussionBoard`

```
public bool EnableDiscussionBoard { set; get; }
```

- `EnableDistributeToSite`

```
public bool EnableDistributeToSite { set; get; }
```

- `EnableDocumentsInNotifications`. If true, all notification emails on asset types will have the asset added as an attachment.

```
public bool EnableDocumentsInNotifications { set; get; }
```

- `EnableGroupEmail`. If true, the community group will accept email using the email address.

```
public bool EnableGroupEmail { set; get; }
```

- `EnableTodoList`

```
public bool EnableTodoList { set; get; }
```

- `Enroll`

```
public bool Enroll { set; get; }
```

- `FolderId`

```
public long FolderId { set; get; }
```

- `GroupAdmin`

```
public Ektron.Cms.UserBaseData GroupAdmin { set; get; }
```

- `GroupCategory`

```
public string GroupCategory { set; get; }
```

- `GroupCreatedDate`

```
public System.DateTime GroupCreatedDate { set; get; }
```

- `GroupEnableDistributeToSite`

```
public bool GroupEnableDistributeToSite { set; get; }
```

- `GroupEnroll`

```
public bool GroupEnroll { set; get; }
```

- `GroupHidden`

```
public bool GroupHidden { set; get; }
```

- `GroupId`. This property is obsolete in 8.5. Use Id.

```
public long GroupId { set; get; }
```

- `GroupImage`

```
public string GroupImage { set; get; }
```

- `GroupLevel`

```
public int GroupLevel { set; get; }
```

- `GroupLocation`

```
public string GroupLocation { set; get; }
```

- GroupLongDescription

```
public string GroupLongDescription { set; get; }
```

- GroupName

```
public string GroupName { set; get; }
```

- GroupParentId

```
public long GroupParentId { set; get; }
```

- GroupPath

```
public string GroupPath { set; get; }
```

- GroupShortDescription

```
public string GroupShortDescription { set; get; }
```

- HasSubGroup

```
public bool HasSubGroup { set; get; }
```

- Hidden

```
public bool Hidden { set; get; }
```

- Id

```
public override long Id { set; get; }
```

- Image

```
public string Image { set; get; }
```

- Level

```
public int Level { set; get; }
```

- Location

```
public string Location { set; get; }
```

- LongDescription

```
public string LongDescription { set; get; }
```

- Name

```
public string Name { set; get; }
```

- ParentId

```
public long ParentId { set; get; }
```

- Path

```
public string Path { set; get; }
```

- ShortDescription

```
public string ShortDescription { set; get; }
```

- Tags. This property is read-only and any changes will not be saved. To add tags to a CommunityGroup, use the AddTagToCommunityGroup API.

```
public string Tags { set; get; }
```

- TotalMember. Gets the total number of member sin the community group. This property is read-only and will not be persisted on Add or Update.

```
public int TotalMember { set; get; }
```

- `ToXmlString()`

```
public string ToXmlString()
```

FavoriteManager

8.50 and higher

The FavoriteManager class manages a list of Ektron CMS content and external Web links (URLs) that a user has designated as favorite content.

Namespace

```
Ektron.Cms.Framework.Community.FavoriteManager
```

Constructors

- `FavoriteManager()`
- `FavoriteManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `FavoriteManagerService`. Returns an instance of the business objects favorite manager service.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Community.FavoriteItemData)
```

Adds a favorite item, with details from the supplied FavoriteItemData object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- * Object Type
- * Object Value
- * User ID
- Description

Parameters

- `favoriteItemData`. The FavoriteItemData object to add.

Returns

A custom FavoriteItemData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTitlelabel" AssociatedControlID="uxTitle" CssClass="span-4
last" runat="server" Text="* Title:" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFavoriteItemObjectTypeLabel"
AssociatedControlID="uxFavoriteItemObjectType" CssClass="span-4 last" runat="server"
Text="* Object Type:" />
        <asp:DropDownList ID="uxFavoriteItemObjectType" runat="server">
            <asp:ListItem>Content</asp:ListItem>
            <asp:ListItem>Url</asp:ListItem>
        </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* Object Value:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last" runat="server" Text="* User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last" runat="server" Text=" Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxDescription" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            Ektron.Cms.Framework.Community.FavoriteManager favoriteManager = new
Ektron.Cms.Framework.Community.FavoriteManager();
            FavoriteItemData favoriteItemData = new FavoriteItemData()
            {
                Title = uxTitle.Text,
                ObjectType = (uxFavoriteItemObjectType.SelectedItem.Text == "Content" ?
Ektron.Cms.Common.FavoriteType.Content : Ektron.Cms.Common.FavoriteType.Url),
                ObjectId = (uxFavoriteItemObjectType.SelectedItem.Text == "Content" ?
long.Parse(uxObjectValue.Text) : 0),
                Url = uxObjectValue.Text,
                Description = uxDescription.Text,
                // TaxonomyId= 334,
                UserId = long.Parse(uxUserId.Text)
            };

            if (uxFavoriteItemObjectType.SelectedItem.Text != "Content")
            {
                FavoriteItemCriteria criteria = new
Ektron.Cms.Community.FavoriteItemCriteria(FavoriteItemProperty.id,
EkEnumeration.OrderByDirection.Ascending);
                criteria.AddFilter(FavoriteItemProperty.Title,
```

```
CriteriaFilterOperator.EqualTo, uxTitle.Text);
                criteria.AddFilter(FavoriteItemProperty.Url,
CriteriaFilterOperator.EqualTo, uxObjectValue.Text);
                List<FavoriteItemData> uxFavoriteItemList = favoriteManager.GetList
(criteria);
                if (uxFavoriteItemList.Count == 0)
                    favoriteManager.Add(favoriteItemData);
                else
                    MessageUtilities.UpdateMessage(uxMessage, "Favorite Name and
Favorite Uri already exists", Message.DisplayModes.Success);
            }
            else
                favoriteManager.Add(favoriteItemData);

            if (string.IsNullOrEmpty(uxMessage.Text))
            {
                if (uxFavoriteItemObjectType.SelectedItem.Text == "Content")
                    MessageUtilities.UpdateMessage(uxMessage, "Favorite Added ",
Message.DisplayModes.Success);
                else
                    MessageUtilities.UpdateMessage(uxMessage, "Favorite Added with Id "
+ favoriteItemData.Id, Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a favorite item from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Object Type

Parameters

- `id`. ID of the favorite item to delete.
- `type`. Favorite type.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFavoriteIdLabel" AssociatedControlID="uxFavoriteId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxFavoriteId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFavoriteItemObjectTypeLabel"
AssociatedControlID="uxFavoriteItemObjectType" CssClass="span-4 last" runat="server"
Text="* Object Type:" />
        <asp:DropDownList ID="uxFavoriteItemObjectType" runat="server">
            <asp:ListItem>Content</asp:ListItem>
            <asp:ListItem>Url</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long favoriteId = long.Parse(uxFavoriteId.Text);
            FavoriteType favoriteType = (uxFavoriteItemObjectType.SelectedItem.Text ==
```

```
"Content" ? Ektron.Cms.Common.FavoriteType.Content :
Ektron.Cms.Common.FavoriteType.Url);

            FavoriteManager favoriteManager = new FavoriteManager();
            favoriteManager.Delete(favoriteId, favoriteType);

            MessageUtilities.UpdateMessage(uxMessage, "Favorite deleted.",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific FavoriteItemData object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Object Type

Parameters

- `id`. ID of the favorite item to get.
- `type`. Favorite type.

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxFavoriteIdLabel" AssociatedControlID="uxFavoriteId"
CssClass="span-4 last" runat="server" Text="* Id:" />
            <ektronUI:TextField ID="uxFavoriteId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
        </li>
        <li class="clearfix" style="color: red;">
            <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
```

```
                Text="Status:" />
        </li>
        <li class="clearfix">
            <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
            <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server"
Text="* - Required" />
        </li>
    </ol>

    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserId" runat="server"></asp:Literal>
    </li>
        <li class="clearfix">
        <asp:Literal ID="uxFavoriteFolderId" runat="server"></asp:Literal>
    </li>
        <li class="clearfix">
        <asp:Literal ID="uxObjectId" runat="server"></asp:Literal>
    </li>
        <li class="clearfix">
        <asp:Literal ID="uxObjectType" runat="server"></asp:Literal>
    </li>
        <li class="clearfix">
        <asp:Literal ID="uxUrl" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long favoriteId = long.Parse(uxFavoriteId.Text);
                        FavoriteType favoriteType = (uxFavoriteItemObjectType.SelectedItem.Text == "C
? Ektron.Cms.Common.FavoriteType.Content : Ektron.Cms.Common.FavoriteType.Url);
```

```
            Ektron.Cms.Framework.Community.FavoriteManager favoriteManager = new
Ektron.Cms.Framework.Community.FavoriteManager();
            FavoriteItemData favoriteItemData = favoriteManager.GetItem(favoriteId,
favoriteType);
            if (favoriteItemData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for favorite folder
with ID " + favoriteItemData.Id.ToString() + " are below",
Message.DisplayModes.Success);
                uxTitle.Text = "Title : " + favoriteItemData.Title;
                uxUserId.Text = "User Id : " + favoriteItemData.UserId;
                uxFavoriteFolderId.Text = "Taxonomy Id : " +
favoriteItemData.TaxonomyId;
                uxObjectId.Text = "ObjectId: " + favoriteItemData.ObjectId;
                uxObjectType.Text = "ObjectType : " + favoriteItemData.ObjectType;
                uxUrl.Text = "Url : " + favoriteItemData.Url;
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Favorite
Id.", Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(FavoriteItemCriteria)
```

Retrieves lists of objects through the GetList(FavoriteItemCriteria) method.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Favorite Item Property
- * Object Value

Parameters

- `criteria`. Criteria used to retrieve objects

Returns

A list of FavoriteItemData.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFavoriteItemPropertyLabel"
AssociatedControlID="uxFavoriteItemProperty"
            CssClass="span-6 last" runat="server" Text="FavoriteItem Property:" />
        <asp:DropDownList ID="uxFavoriteItemProperty" runat="server">
            <asp:ListItem>UserId</asp:ListItem>
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>FavoriteFolderId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last"
            runat="server" Text="ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxFavoriteItemListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
```

```
                        FavoriteTaxonomyId
                    </th>
                    <th>
                        ObjectId
                    </th>
                    <th>
                        ObjectType
                    </th>
                    <th>
                        UserId
                    </th>
                    <th>
                        Url
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TaxonomyId ")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectType")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Url")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            Ektron.Cms.Framework.Community.FavoriteManager favoriteManager = new
Ektron.Cms.Framework.Community.FavoriteManager();
            FavoriteItemCriteria criteria = new
Ektron.Cms.Community.FavoriteItemCriteria(FavoriteItemProperty.id,
EkEnumeration.OrderByDirection.Ascending);

            object Objectvalue;
            string Property = uxFavoriteItemProperty.SelectedItem.Text;
            Objectvalue = Convert.ToInt64(uxObjectValue.Text);

            if (Property == "Id")
            {
                criteria.AddFilter(FavoriteItemProperty.id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (Property == "FavoriteTaxonomyId")
            {
                criteria.AddFilter(FavoriteItemProperty.FavoriteTaxonomyId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                criteria.AddFilter(FavoriteItemProperty.UserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            List<FavoriteItemData> uxFavoriteItemList = favoriteManager.GetList
(criteria);

            uxFavoriteItemListView.Visible = true;
            uxFavoriteItemListView.DataSource = uxFavoriteItemList;
            uxFavoriteItemListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.FavoriteItemData)
```

Update an existing favorite item in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Title
- * Object Type
- * Object Value
- Description

Parameters

- `favoriteItemData`. The FavoriteItemData object to update.

Returns

Returns the custom FavoriteItemData object that was updated.

Remarks

Validate the Favorite item with `GetItem()` before you update the properties. Then, call `Update` with your modified FavoriteItemData object.

NOTE: You cannot change all properties after the initial Add event, such as the FavoriteItemData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFavoriteIdLabel" AssociatedControlID="uxFavoriteId"
CssClass="span-4 last" runat="server" Text="*Id:" />
        <ektronUI:TextField ID="uxFavoriteId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTitlelabel" AssociatedControlID="uxTitle" CssClass="span-4
last" runat="server" Text="* Title:" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFavoriteItemObjectTypeLabel"
AssociatedControlID="uxFavoriteItemObjectType" CssClass="span-4 last" runat="server"
```

```
Text="* Object Type:" />
        <asp:DropDownList ID="uxFavoriteItemObjectType" runat="server">
            <asp:ListItem>Url</asp:ListItem>
        </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* Object Value:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last" runat="server" Text=" Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxDescription" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            Ektron.Cms.Framework.Community.FavoriteManager favoriteManager = new
Ektron.Cms.Framework.Community.FavoriteManager();
            FavoriteItemData favoriteItemData = new FavoriteItemData();

            long favoriteId = long.Parse(uxFavoriteId.Text);

            favoriteItemData = favoriteManager.GetItem(favoriteId,
Ektron.Cms.Common.FavoriteType.Url);
```

```
            if (favoriteItemData != null)
            {
                favoriteItemData.Title = uxTitle.Text != string.Empty ? uxTitle.Text :
favoriteItemData.Title;
                favoriteItemData.Description = uxDescription.Text != string.Empty ?
uxDescription.Text : favoriteItemData.Description;
                favoriteItemData.Url = uxObjectValue.Text != string.Empty ?
uxObjectValue.Text : favoriteItemData.Url;
                favoriteItemData.ObjectType = Ektron.Cms.Common.FavoriteType.Url;
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Favorite
Id.", Message.DisplayModes.Error);
                return;
            }

            favoriteManager.Update(favoriteItemData);
            MessageUtilities.UpdateMessage(uxMessage, "Favorite Updated",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

FavoriteItemCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Community
```

Constructors

- `FavoriteItemCriteria()`

  ```
  public FavoriteItemCriteria()
  ```

- `FavoriteItemCriteria(Ektron.Cms.Common.FavoriteItemProperty, EkEnumeration.OrderByDirection)`

  ```
  public FavoriteItemCriteria(Ektron.Cms.Common.FavoriteItemProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `FavoriteItemProperty` are:

- Description
- FavoriteTaxonomyId
- FavoriteType
- Id
- ObjectId
- Title
- Url
- UserId

FavoriteItemData

Namespace

```
Ektron.Cms.Community
```

Properties

- `Description`. Gets or sets the description of the favorite.

  ```
  public string Description { set; get; }
  ```

- `Id`. Gets or sets the ID of the CMS Data object. This property is inherited from the CMSData base class and is not applicable to FavoriteItemData. FavoriteItemData does not have a unique ID.

  ```
  public override long Id { set; get; }
  ```

- `ObjectId`. Gets or sets the Id of the favorite object. This would be the content ID if the favorite type is Content.

  ```
  public long ObjectId { set; get; }
  ```

- `ObjectType`. Gets or sets the Favorite type.

  ```
  public Ektron.Cms.Common.FavoriteType ObjectType { set; get; }
  ```

- `TaxonomyId`. Gets or sets the ID of the Taxonomy Node this favorite Item saved in.

  ```
  public long TaxonomyId { set; get; }
  ```

- `Title`. Gets or sets the Title of the favorite. If its a content favorite, this is the content title and cannot be changed. If its a URL favorite, then it can be set.

  ```
  public string Title { set; get; }
  ```

- `Url`. Gets or sets the URL of the favorite.

  ```
  public string Url { set; get; }
  ```

- `UserId`. Gets or setsa the User ID whom this favorite Item belongs to.

  ```
  public long UserId { set; get; }
  ```

FavoriteTaxonomyManager

8.50 and higher

The FavoriteTaxonomyManager class manages user favorite taxonomies.

Ektron's taxonomy is a content-level categorization system that uses one-to-many relationships to create a scalable organization of content. Content is categorized in the database by how it relates to multiple categories, allowing it to be accessed in multiple ways for multiple purposes. The hierarchy of the content is arranged from the general to the specific, and there may be multiple "routes" that define the content at the end of the hierarchy.

Namespace

```
Ektron.Cms.Framework.Community.FavoriteTaxonomyManager
```

Constructors

- `FavoriteTaxonomyManager()`
- `FavoriteTaxonomyManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `FavoriteFolderManagerService`. Returns an instance of the business objects task manager service.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* on the facing page
- *Delete* on page 885
- *GetItem* on page 886
- *GetTree* on page 888

- *GetUserTaxonomy* on page 890
- *Update* on page 892

Add

```
Add(Ektron.Cms.Community.FavoriteTaxonomyData)
```

Adds a favorite taxonomy.

When you add the FavoriteTaxonomyData object, the method returns a favorite taxonomy ID object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- * Parent ID (Taxonomy ID)
- Description

Parameters

- `favoriteTaxonomyData`. The FavoriteTaxonomyData object to add

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTitlelabel" AssociatedControlID="uxTitle" CssClass="span-4
last" runat="server" Text="* Title:" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="Favorites"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxParentIdLabel" AssociatedControlID="uxParentId"
CssClass="span-4 last" runat="server" Text="* Parent Id(Taxonomy Id):" />
        <ektronUI:TextField ID="uxParentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"/>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last" runat="server" Text=" Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxDescription" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
```

```
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            FavoriteTaxonomyManager favoritetaxonomyManager = new
FavoriteTaxonomyManager();

            long taxonomyId = long.Parse(uxParentId.Text);
            PagingInfo pagingInfo = new PagingInfo();
            pagingInfo.RecordsPerPage = 100;

            Ektron.Cms.Community.FavoriteTaxonomyData favoriteTaxonomyData =
favoritetaxonomyManager.GetTree(taxonomyId, 3, true, pagingInfo);

            if (favoriteTaxonomyData != null)
            {
                Ektron.Cms.Community.FavoriteTaxonomyData favoriteTaxonomyData1 = new
Ektron.Cms.Community.FavoriteTaxonomyData()
                {
                    Name = uxTitle.Text,
                    ParentId = taxonomyId,
                    Description = uxDescription.Text,
                };

                favoritetaxonomyManager.Add(favoriteTaxonomyData1);
                MessageUtilities.UpdateMessage(uxMessage, "Favorite taxonomy Added with
Id " + favoriteTaxonomyData.Id, Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "please enter valid parent id
(taxonomy id).", Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a favorite taxonomy from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The favorite taxonomy ID to be deleted.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFavoriteFolderIdLabel"
AssociatedControlID="uxFavoriteFolderId" CssClass="span-3 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField ID="uxFavoriteFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long favoriteTaxonomyId = long.Parse(uxFavoriteFolderId.Text);

            FavoriteTaxonomyManager favoritetaxonomyManager = new
FavoriteTaxonomyManager();
            favoritetaxonomyManager.Delete(favoriteTaxonomyId);

            MessageUtilities.UpdateMessage(uxMessage, "Favorite taxonomy deleted.",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific TaxonomyData object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the favorite taxonomy to retrieve. Internally the folder is stored as Taxonomy so this is the corresponding taxonomy ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFavoriteFolderIdLabel"
AssociatedControlID="uxFavoriteFolderId" CssClass="span-5 last" runat="server" Text="*
Id :" />
        <ektronUI:TextField ID="uxFavoriteFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxParentId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long favoriteTaxonomyId = long.Parse(uxFavoriteFolderId.Text);

            FavoriteTaxonomyManager favoriteTaxonomyManager = new
FavoriteTaxonomyManager();
```

```
            Ektron.Cms.Community.FavoriteTaxonomyData favoriteTaxonomyData =
favoriteTaxonomyManager.GetItem(favoriteTaxonomyId);

            if (favoriteTaxonomyData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Details for favorite
taxonomy with ID " + favoriteTaxonomyData.Id.ToString() + " are below",
Message.DisplayModes.Success);
                uxName.Text = "Name : " + favoriteTaxonomyData.Name;
                uxParentId.Text = "Parent Id : " + favoriteTaxonomyData.ParentId;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Favorite taxonomy does not
exist. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTree

```
GetTree(System.Int64,System.Int32,System.Boolean,Ektron.Cms.PagingInfo)
```

Retrieves the taxonomy tree for a favorite.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Depth
- * Include Items
- * Records Per Page

Parameters

- `id`. The FavoriteTaxonomyData ID to be retrieved. Internally the folder is stored as Taxonomy so this is the corresponding taxonomy ID.
- `depth`. The FavoriteTaxonomyData structure depth to retrieve.

- `includeItems`. Set whether the returned object contains included items.
- `pagingInfo`. Set paging details.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFavoriteFolderIdLabel"
AssociatedControlID="uxFavoriteFolderId" CssClass="span-5 last" runat="server" Text="*
Id :" />
        <ektronUI:TextField ID="uxFavoriteFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDepthLabel" AssociatedControlID="uxDepth" CssClass="span-5
last" runat="server" Text="* Depth :" />
        <ektronUI:TextField ID="uxDepth" CssClass="span-6" Text="3" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIncludeItemsLabel" AssociatedControlID="uxIncludeItems"
CssClass="span-5 last" runat="server" Text="* Include Items:" />
        <ektronUI:Button DisplayMode="Checkbox"  Text="" ID="uxIncludeItems"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRecordsPerPageLabel1"
AssociatedControlID="uxRecordsPerPage" CssClass="span-5 last" runat="server" Text="*
Records Per Page :" />
        <ektronUI:TextField ID="uxRecordsPerPage" CssClass="span-6" Text="100"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxParentId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long favoriteTaxonomyId = long.Parse(uxFavoriteFolderId.Text);

            PagingInfo pagingInfo = new PagingInfo();
            pagingInfo.RecordsPerPage = int.Parse(uxRecordsPerPage.Text);

            FavoriteTaxonomyManager favoriteTaxonomyManager = new
FavoriteTaxonomyManager();
            Ektron.Cms.Community.FavoriteTaxonomyData favoriteTaxonomyData =
favoriteTaxonomyManager.GetTree(favoriteTaxonomyId, int.Parse(uxDepth.Text),
uxIncludeItems.Checked, pagingInfo);

            if (favoriteTaxonomyData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for favorite taxonomy
with ID " + favoriteTaxonomyData.Id.ToString() + " are below.",
Message.DisplayModes.Success);
                uxName.Text = "Name : " + favoriteTaxonomyData.Name;
                uxParentId.Text = "Parent Id : " + favoriteTaxonomyData.ParentId;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Favorite taxonomy does not
exist. ", Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetUserTaxonomy

```
GetUserTaxonomy(System.Int64)
```

Retrieves the taxonomy tree for a favorite.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `userId`. The user ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last" runat="server" Text="* User Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
User Taxonomy"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxParentId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
```

```
        {
            long userId = long.Parse(uxUserId.Text);

            FavoriteTaxonomyManager favoriteTaxonomyrManager = new
FavoriteTaxonomyManager();
            Ektron.Cms.Community.FavoriteTaxonomyData favoriteTaxonomyData =
favoriteTaxonomyrManager.GetUserTaxonomy(userId);

            if (favoriteTaxonomyData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Details for favorite
taxonomy with ID " + favoriteTaxonomyData.Id.ToString() + " are below",
Message.DisplayModes.Success);
                uxName.Text = "Name : " + favoriteTaxonomyData.Name;
                uxParentId.Text = "Parent Id : " + favoriteTaxonomyData.ParentId;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Favorite taxonomy does not
exist. ", Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.FavoriteTaxonomyData)
```

Updates an existing favorite taxonomy item in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Favorite Taxonomy ID
- Title
- Description

Parameters

- `favoriteTaxonomyData`. The FavoriteTaxonomyData object to update.

Remarks

Validate the favorite taxonomy item with `GetItem()` before you update the properties. Then, call `Update` with your modified FavoriteTaxonomyData object.

NOTE: You cannot change all properties after the initial Add event, such as the FavoriteTaxonomyData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFavoriteFolderIdLabel"
AssociatedControlID="uxFavoriteFolderId" CssClass="span-4 last" runat="server" Text="*
Favorite taxonomy Id:" />
        <ektronUI:TextField ID="uxFavoriteFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTitlelabel" AssociatedControlID="uxTitle" CssClass="span-4
last" runat="server" Text="Title:" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last" runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxDescription" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
```

```
        {
            long favoriteFolderId = long.Parse(uxFavoriteFolderId.Text);

            PagingInfo pagingInfo = new PagingInfo();
            pagingInfo.RecordsPerPage = 100;

            FavoriteTaxonomyManager favoriteTaxonomyManager = new
FavoriteTaxonomyManager();
            Ektron.Cms.Community.FavoriteTaxonomyData favoriteTaxonomyData =
favoriteTaxonomyManager.GetTree(favoriteFolderId, 3, true, pagingInfo);

            if (favoriteTaxonomyData != null)
            {
                favoriteTaxonomyData.Name = uxTitle.Text != string.Empty ? uxTitle.Text
: favoriteTaxonomyData.Name;
                favoriteTaxonomyData.Description = uxDescription.Text != string.Empty ?
uxDescription.Text : favoriteTaxonomyData.Description;

                favoriteTaxonomyManager.Update(favoriteTaxonomyData);

                MessageUtilities.UpdateMessage(uxMessage, "Favorite folder updated",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Favorite folder does not
exist. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

FavoriteTaxonomyData

Namespace

```
Ektron.Cms.Community
```

Properties

- `ChildTaxonomies`. Gets or sets the favorite child taxonomy in the current taxonomy. If not retrieved, this will be an empty list.

  ```
  public System.Collections.Generic.List<FavoriteTaxonomyData>
    ChildTaxonomies { set; get; }
  ```

- `Description`. Gets or sets the description of the favorites's taxonomy.

```
public string Description { set; get; }
```

- `FavoriteTaxonomyData()`

```
public FavoriteTaxonomyData()
```

- `Id`. Gets or sets the ID of the favorite taxonomy. Overridden to clarify the description and taxonomy relationship

```
public override long Id { set; get; }
```

- `Items`. Gets or sets the Favorites in the taxonomy. If not retrieved, this will be an empty list.

```
public System.Collections.Generic.List<FriendsData> Items { set; get; }
```

- `Name`. Gets or sets the Favorites taxonomy name.

```
public string Name { set; get; }
```

- `ParentId`. Gets or sets the ID of the Favorites folder's parent taxonomy.

```
public long ParentId { set; get; }
```

FlagManager

8.50 and higher

The FlagManager class manages flags that are assigned to content to provide feedback. For information about flagging, see Flagging in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Community.FlagManager
```

Constructors

- `FlagManager()`
- `FlagManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `FlagManagerService`. Returns an instance of the business objects FlagManager service.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.ObjectFlagData)
```

Adds a flag to community content whose ID is specified in the objectFlagData.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object Type
- * Object ID
- * Flag Option
- Flag Comment

Parameters

- `objectFlagData`. The ObjectFlagData object.

Remarks

Before using this add method, define a flag. Do this by logging into the CMS and navigating to **Workarea > Settings > Community Management > Flagging > Community Flagging Definition**.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFlagObjectTypeLabel"
AssociatedControlID="uxFlagObjectType" CssClass="span-4 last" runat="server" Text="*
Object Type:" />
        <asp:DropDownList ID="uxFlagObjectType" runat="server">
            <asp:ListItem>Content</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* Object Id:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30"  />
    </li>
    <li class="clearfix">
                <ektronUI:Label ID="uxFlagOptionLabel"
AssociatedControlID="uxFlagOption" CssClass="span-6 last" runat="server" Text="* Flag
Option:" />
                <asp:DropDownList ID="uxFlagOption" runat="server">
                    <asp:ListItem Value="1">Inappropriate</asp:ListItem>
                    <asp:ListItem Value="2">OffensiveContent</asp:ListItem>
                    <asp:ListItem Value="3">CopyrightViolation</asp:ListItem>
                </asp:DropDownList>
            </li>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxFlagCommentLabel" AssociatedControlID="uxFlagComment"
CssClass="span-4 last" runat="server" Text=" Flag Comment:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxFlagComment" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            string visitorId = string.Empty;
            if (Page.Request.Cookies["ekContentRatingID"] == null)
            {
                visitorId = (string)(System.Guid.NewGuid().ToString().Replace("-", ""));
                Page.Response.Cookies.Add(new System.Web.HttpCookie("ekContentRatingID",
visitorId));
                Page.Response.Cookies["ekContentRatingID"].Expires = DateTime.MaxValue;
            }
            else
            {
                visitorId = Page.Request.Cookies["ekContentRatingID"].Value;
            }
            ContentManager contentManager = new ContentManager();
            ContentData cData = new ContentData();
            long contentID = long.Parse(uxObjectValue.Text);

            cData = contentManager.GetItem(contentID, false);
            if (cData != null && cData.IsPublished)
```

```
            {

                FlagManager flagManager = new FlagManager();

                long option = 1;
                option = long.Parse(uxFlagOption.SelectedValue);


                ObjectFlagData objectFlagData = new ObjectFlagData()
                {
                    ObjectType = Ektron.Cms.Common.EkEnumeration.CMSObjectTypes.Content,
                    ObjectId = contentID,
                    UserId = flagManager.RequestInformation.UserId,
                    VisitorID = visitorId,
                    Comment  = uxFlagComment.Text,
                    FlagOptionId = option,
                };

                flagManager.Add(objectFlagData);
                MessageUtilities.UpdateMessage(uxMessage, "Content Flag Added with Id "
+ objectFlagData.Id, Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Content
ID.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a flag from community content whose ID is specified in the objectFlagData.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object Flag ID

Parameters

- `objectFlagId`. The ID of the ObjectFlagData type to be deleted.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectFlagIdLabel" AssociatedControlID="uxObjectFlagId"
CssClass="span-4 last" runat="server" Text="* Object Flag Id:" />
        <ektronUI:TextField ID="uxObjectFlagId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long objectFlagId = long.Parse(uxObjectFlagId.Text);

            FlagManager flagManager = new FlagManager();
            ObjectFlagData objectFlagData = flagManager.GetItem(objectFlagId);

            if (objectFlagData != null)
            {
```

```
                flagManager.Delete(objectFlagId);
                MessageUtilities.UpdateMessage(uxMessage, "Flag deleted.",
Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus,  "Please Enter Valid Flag ID.",
Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific FlagData object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `objectFlagId`. The ID of the ObjectFlagData type to get.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectFlagIdLabel" AssociatedControlID="uxObjectFlagId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxObjectFlagId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
```

```
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxObjectType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxObjectId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFlagComment" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFlagDate" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long objectFlagId = long.Parse(uxObjectFlagId.Text);

            FlagManager flagManager = new FlagManager();
            ObjectFlagData objectFlagData = flagManager.GetItem(objectFlagId);

            if (objectFlagData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Details for Flag with ID " +
objectFlagId.ToString() + " are below", Message.DisplayModes.Success);
                uxObjectType.Text = "Object Type : " +
(Ektron.Cms.Common.EkEnumeration.CMSObjectTypes)objectFlagData.ObjectType;
                uxObjectId.Text = "Object Id : " + objectFlagData.ObjectId;
```

```
                uxUserId.Text = "User Id : " + objectFlagData.UserId;
                uxFlagComment.Text = "Flag Comment : " + objectFlagData.Comment;
                uxFlagDate.Text = "Flag Date :" + objectFlagData.Date;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Flag does not exists. ",
Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Community.FlagCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Object Flag Data Property
- * Object Value

Parameters

- `criteria`. *FlagCriteria* on page 908 used to retrieve content.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectFlagDataPropertyLabel"
AssociatedControlID="uxObjectFlagDataProperty" CssClass="span-6 last" runat="server"
Text="Object Flag Data Property:" />
        <asp:DropDownList ID="uxObjectFlagDataProperty" runat="server">
            <asp:ListItem>Flag Entry Id</asp:ListItem>
            <asp:ListItem>Object Id</asp:ListItem>
```

```
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxObjectFlagListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Flag Entry Id
                    </th>
                    <th>
                        Object Id
                    </th>
                    <th>
                        Flag Comment
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("FlagEntryId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FlagComment")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            Ektron.Cms.Community.FlagCriteria criteria = new
Ektron.Cms.Community.FlagCriteria();
            if (uxObjectFlagDataProperty.SelectedItem.Text == "Flag Entry Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(ObjectFlagDataProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(ObjectFlagDataProperty.ObjectId ,
CriteriaFilterOperator.EqualTo , Objectvalue);
            }

            FlagManager flagManager = new FlagManager();
            List<ObjectFlagData> objectFlagDataList = flagManager.GetList(criteria);

            uxObjectFlagListView.Visible = true;
            uxObjectFlagListView.DataSource = objectFlagDataList;
            uxObjectFlagListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.ObjectFlagData)
```

Updates an existing ObjectFlag item in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- Flag Option
- Flag Comment

Parameters

- `objectFlagData`. The flag object.

Returns

Returns the custom ObjectFlagData object updated.

Remarks

Validate the flag item with `GetItem()` before you update the properties. Then, call `Update` with your modified ObjectFlagData object.

NOTE: You cannot change all properties after the initial Add event, such as the ObjectFlagData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectFlagIdLabel" AssociatedControlID="uxObjectFlagId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxObjectFlagId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxFlagItemIdLabel" AssociatedControlID="uxFlagItemId"
CssClass="span-3 last" runat="server" Text=" Flag Item Id:" />
        <ektronUI:TextField ID="uxFlagItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFlagCommentLabel" AssociatedControlID="uxFlagComment"
CssClass="span-3 last" runat="server" Text=" Flag Comment:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxFlagComment" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
```

```
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long objectFlagId = long.Parse(uxObjectFlagId.Text);

            FlagManager flagManager = new FlagManager();
            ObjectFlagData objectFlagData = flagManager.GetItem(objectFlagId);

            if (objectFlagData != null)
            {
                long option = 0;
                option = long.Parse(uxFlagOption.SelectedValue);
                if (option > 0)
                {
                    objectFlagData.FlagOptionId = option;
                }
                objectFlagData.Comment  = (uxFlagComment.Text != string.Empty ?
uxFlagComment.Text : objectFlagData.Comment);

                flagManager.Update(objectFlagData);

                MessageUtilities.UpdateMessage(uxMessage, "Content Flag Updated ",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Flag does not exists. ",
Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

FlagCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Community
```

Constructors

- `FlagCriteria()`

  ```
  public FlagCriteria()
  ```

- `FlagCriteria(Ektron.Cms.Common.FlagProperty, EkEnumeration.OrderByDirection)`

  ```
  public FlagCriteria(Ektron.Cms.Common.FlagProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `FlagProperty` are:

  - Comment
  - Date
  - FlagDefinitionId
  - FlagOptionId
  - Id
  - ObjectId
  - ObjectLanguageId
  - ObjectType
  - UserId
  - VisitorId

ObjectFlagData

Namespace

```
Ektron.Cms
```

Properties

- `Comment`. Gets or sets the comment associated with this flag instance.

  ```
  public string Comment { set; get; }
  ```

- `Date`. Gets or sets the Date the object was flagged.

  ```
  public System.DateTime Date { set; get; }
  ```

- FlagComment

```
public string FlagComment { set; get; }
```

- FlagDate

```
public System.DateTime FlagDate { set; get; }
```

- FlagDefinitionId. Gets the ID of the Flag Definition used by this flag instance. When saving this value is not neccessary and will be derived from FlagOptionId.

```
public long FlagDefinitionId { set; get; }
```

- FlagEntryId

```
public long FlagEntryId { set; get; }
```

- FlagId

```
public long FlagId { set; get; }
```

- FlagItemId

```
public long FlagItemId { set; get; }
```

- FlagOptionId. Gets or sets the ID of the Flag Definition Option that is being chosen for this Flag instance.

```
public long FlagOptionId { set; get; }
```

- Id. Gets the unique ID of this Object Flag Instance once saved.

```
public override long Id { set; get; }
```

- ObjectId. Gets or sets the ID of the object that is being flagged. This property correlates directly with ObjectType. For Example, if ObjectType == Content, then ObjectId would be the Content ID being flagged.

```
public long ObjectId { set; get; }
```

- ObjectLanguageId. Gets or sets the Id of the Language of the object being flagged.

```
public int ObjectLanguageId { set; get; }
```

- ObjectType. Gets or sets the Type of Object being flagged.

```
public EkEnumeration.CMSObjectTypes ObjectType { set; get; }
```

- ToXmlString()

```
public string ToXmlString()
```

- UserId. Gets or sets the Id of the logged in user who is creting this flag instance. If the user is not logged in, use visitorId.

```
public long UserId { set; get; }
```

- VisitorId. Gets or sets the ID of the non-logged in user that performed this flag. This property is only used for users that are not logged in. The CMS trackes these users by a unique visitor ID stored in a cookie and may overwrite value of this property. Logged in users should populate the UserId property.

```
public string VisitorID { set; get; }
```

FriendsManager

8.50 and higher

The FriendsManager class manages colleague connections (friends).

Namespace

```
Ektron.Cms.Framework.Community.FriendsManager
```

Constructors

- `FriendsManager()`
- `FriendsManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `FriendsManagerService`. Returns an instance of the business objects friends manager service.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *GetPendingList* on page 928
- *Invite (email)* on page 930 (email)
- *Invite (user)* on page 931 (user)
- *IsFriend* on page 933
- *Update* on page 934

AcceptInvite

```
AcceptInvite(Ektron.Cms.Community.PendingFriendsData)
```

Accepts a friend's invitation.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Pending Friend ID

Parameters

- `pendingFriendData`. *PendingFriendsData* on page 938 details.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendIdLabel" AssociatedControlID="uxFriendId"
CssClass="span-4 last" runat="server" Text="* Pending Friend Id:" />
        <ektronUI:TextField ID="uxFriendId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Accept Invite"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            friendId = long.Parse(uxFriendId.Text);

            FriendsManager friendsManager = new FriendsManager();
            List<PendingFriendsData> pendingFriendsData = friendsManager.GetPendingList
(friendsManager.RequestInformation.UserId);

            PendingFriendsData selectedFriend = pendingFriendsData.FirstOrDefault(x =>
x.UserId == friendId);
            if (selectedFriend != null)
            {
                friendsManager.AcceptInvite(selectedFriend);
                MessageUtilities.UpdateMessage(uxMessage,"Invite accepted. ",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Invite not found. ",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Add

```
Add(Ektron.Cms.FriendsData)
```

Adds a friends data object. When you add the FriendsData, the method returns a Friends object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Friend User ID

Parameters

- `friendsData`. The FriendsData object to add.

Returns

Returns the custom FriendsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendUserIdlabel" AssociatedControlID="uxFriendUserId"
CssClass="span-4 last" runat="server" Text="* Friend UserId:" />
        <ektronUI:TextField ID="uxFriendUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Framework.User;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            FriendsManager friendsManager = new FriendsManager();
            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxFriendUserId.Text);
            //Check Whether Entered UserId is Valid or Not.
            UserData Userdata = Usermanager.GetItem(UserId);
```

```
            if (Userdata != null)
            {

                FriendsData friendsData = new FriendsData()
                {
                    FriendUserId = UserId,
                    UserId = friendsManager.RequestInformation.UserId
                    //TaxonomyId = 305
                };

                friendsManager.Add(friendsData);
                MessageUtilities.UpdateMessage(uxMessage,  "Friend Added",
Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User ID and
Try again.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeclineInvite

```
DeclineInvite(PendingFriendsData)
```

Declines a friend invitation.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Pending Friend ID

Parameters

- `pendingFriendData`. PendingFriendsData details.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxFriendIdLabel" AssociatedControlID="uxFriendId"
CssClass="span-4 last" runat="server" Text="* Pending Friend Id:" />
        <ektronUI:TextField ID="uxFriendId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Decline Invite"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            friendId = long.Parse(uxFriendId.Text);

            FriendsManager friendsManager = new FriendsManager();
            List<PendingFriendsData> pendingFriendsData = friendsManager.GetPendingList
(friendsManager.RequestInformation.UserId);

            PendingFriendsData selectedFriend = pendingFriendsData.Find(FindFriend);
            if (selectedFriend != null)
            {
                friendsManager.DeclineInvite(selectedFriend);
                MessageUtilities.UpdateMessage(uxMessage, "Invite declined. ",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Invite not found. ",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }

    protected bool FindFriend(PendingFriendsData pendingFriend)
    {
        if (pendingFriend.UserId == friendId)
            return true;
        else
            return false;
    }
```

Delete (friend)

```
Delete(System.Int64)
```

Deletes a friend object from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `friendId`. The ID of the friend to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendIdLabel" AssociatedControlID="uxFriendId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxFriendId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
```

```
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long friendId = long.Parse(uxFriendId.Text);

            FriendsManager friendsManager = new FriendsManager();
            FriendsData friendsData = friendsManager.GetItem(friendId);

            if (friendsData != null)
            {
                friendsManager.Delete(friendId);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Friend
ID.", Message.DisplayModes.Error);
                return;
            }

            MessageUtilities.UpdateMessage(uxMessage, "Friend with Id " + friendId + "
deleted.", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete (user's friend)

```
Delete(System.Int64)
```

Deletes a specified user's Friend.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Friend ID

Parameters

- `friendId`. The ID of the friend to delete.
- `userId`. ID of user deleting friend.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last" runat="server" Text="* User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendIdLabel" AssociatedControlID="uxFriendId"
CssClass="span-3 last" runat="server" Text="* Friend Id:" />
        <ektronUI:TextField ID="uxFriendId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.User;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long userId = long.Parse(uxUserId.Text);
            long friendId = long.Parse(uxFriendId.Text);

            FriendsManager friendsManager = new FriendsManager();
            UserManager Usermanager = new UserManager();
            FriendsData friendsData = friendsManager.GetItem(friendId);

            if (friendsData != null)
            {
                UserData userdata = Usermanager.GetItem(userId);
                if (userdata != null)
                {
                    friendsManager.Delete(friendId, userId);
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User
ID.", Message.DisplayModes.Error);
                    return;
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Friend
ID.", Message.DisplayModes.Error);
                return;
            }

            MessageUtilities.UpdateMessage(uxMessage, "Friend with Id " + friendId + "
deleted.", Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
```

```
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetInvitedList

```
GetInvitedList(System.Int64)
```

Retrieves a list of PendingFriendsData objects that have an Invited state.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `userId`. ID used to retrieve friends.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last" runat="server" Text="* User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxPendingFriendsDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        User Id
                    </th>
```

```
                    <th>
                        InvitationId
                    </th>
                    <th>
                        Friend User Id
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("InvitationId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FriendUserId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long userId = long.Parse(uxUserId.Text);

            FriendsManager friendsManager = new FriendsManager();
            List<PendingFriendsData> pendingFriendsData = friendsManager.GetInvitedList
(userId);

            uxPendingFriendsDataListView.Visible = true;
            uxPendingFriendsDataListView.DataSource = pendingFriendsData;
            uxPendingFriendsDataListView.DataBind();
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem (ID)

```
GetItem(System.Int64)
```

Retrieves the properties of a specific FriendsData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `friendId`. The ID of the user's friend to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendIdLabel" AssociatedControlID="uxFriendId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxFriendId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
```

```
        <asp:Literal ID="uxUserId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long friendId = long.Parse(uxFriendId.Text);

            FriendsManager friendsManager = new FriendsManager();
            FriendsData friendsData = friendsManager.GetItem(friendId);

            if (friendsData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for friend with ID "
+ friendsData.FriendUserId.ToString() + " are below", Message.DisplayModes.Success);
                uxName.Text = "Name : " + friendsData.DisplayName;
                uxUserId.Text = "User Id : " + friendsData.UserId;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Friend does not exists. ",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem (ID,ID)

```
GetItem(System.Int64,System.Int64)
```

Retrieves the properties of a specific FriendsData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Friend ID

Parameters

- `friendId`. The ID of the user's friend to retrieve.
- `userId`. The ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendIdLabel" AssociatedControlID="uxFriendId"
CssClass="span-3 last"
            runat="server" Text="*Friend Id:" />
        <ektronUI:TextField ID="uxFriendId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Message ID="uxStatus" DisplayMode="Error" runat="server"
Visible="false" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long friendId = long.Parse(uxFriendId.Text);
            long userId = long.Parse(uxUserId.Text);

            FriendsManager friendsManager = new FriendsManager();
            FriendsData friendsData = friendsManager.GetItem(friendId, userId);

            if (friendsData != null)
            {
                uxMessage.Text = "Details for friend with ID " +
friendsData.FriendUserId.ToString() + " are below";
                uxName.Text = "Name : " + friendsData.DisplayName;
            }
            else
            {
                uxMessage.Text = "Friend does not exists. ";
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            uxMessage.Text = ex.Message;
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(FriendsCriteria)
```

Retrieves lists of objects through the GetList(FriendsCriteria) method.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Friends Data Property
- * Object Value

Parameters

- `criteria`. Criteria used to retrieve friends.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendsDataPropertyLabel"
AssociatedControlID="uxFriendsDataProperty" CssClass="span-4 last" runat="server"
Text="* FriendsDataProperty:" />
        <asp:DropDownList ID="uxFriendsDataProperty" runat="server">
            <asp:ListItem>FriendUserId</asp:ListItem>
            <asp:ListItem>UserId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxfriendsDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        User Id
                    </th>
                    <th>
                        First Name
                    </th>
                    <th>
                        Friend UserId
                    </th>
                </tr>
            </thead>
            <tbody>
```

```
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FirstName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FriendUserId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            FriendsCriteria criteria = new FriendsCriteria(FriendsProperty.UserId,
EkEnumeration.OrderByDirection.Ascending);
            if (uxFriendsDataProperty.SelectedItem.Text == "FriendUserId")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(FriendsProperty.FriendUserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(FriendsProperty.UserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            FriendsManager friendsManager = new FriendsManager();
```

```
            List<FriendsData> friendsDataList = friendsManager.GetList(criteria);

            uxfriendsDataListView.Visible = true;
            uxfriendsDataListView.DataSource = friendsDataList;
            uxfriendsDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetPendingList

```
GetPendingList(System.Int64)
```

Retrieves a list of PendingFriendsData objects that have a Pending state.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `userId`. ID used to retrieve friends.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last" runat="server" Text="* User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

```
<asp:ListView ID="uxPendingFriendsDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        User Id
                    </th>
                    <th>
                        InvitationId
                    </th>
                    <th>
                        Friend User Id
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("InvitationId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FriendUserId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            long userId = long.Parse(uxUserId.Text );

            FriendsManager friendsManager = new FriendsManager();
            List<PendingFriendsData> pendingFriendsData = friendsManager.GetPendingList
(userId);

            uxPendingFriendsDataListView.Visible = true;
            uxPendingFriendsDataListView.DataSource = pendingFriendsData;
            uxPendingFriendsDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Invite (email)

```
Invite(System.Int64,System.String,System.String)
```

Invites a user to be your friend.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Friend ID
- * Subject
- * Message

Parameters

- `friendId`. User ID of person to invite as a friend.
- `subject`. Subject of the invitation to be sent.
- `message`. Message of the invitation to be sent.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendIdLabel" AssociatedControlID="uxFriendId"
CssClass="span-4 last" runat="server" Text="* Friend Id:" />
        <ektronUI:TextField ID="uxFriendId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxSubjectlabel" AssociatedControlID="uxSubject"
CssClass="span-4 last" runat="server" Text="* Subject:" />
        <ektronUI:TextField ID="uxSubject" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxMessageLabel" AssociatedControlID="uxInviteMessage"
CssClass="span-4 last" runat="server" Text="* Message:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxInviteMessage" CssClass="span-
6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Invite"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long friendId = long.Parse(uxFriendId.Text);

            FriendsManager friendsManager = new FriendsManager();
            friendsManager.Invite(friendId, uxSubject.Text, uxInviteMessage.Text);

            MessageUtilities.UpdateMessage(uxMessage, "Invite sent. ",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Invite (user)

```
Invite(Ektron.Cms.Common.invitationSendRequestData)
```

8.60 and higher

Invites a user to be your friend.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Friend ID

Parameters

- `invitationSendRequestData`. InvitationSendRequestData.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendIdLabel" AssociatedControlID="uxFriendId"
CssClass="span-4 last" runat="server" Text="* Friend Id:" />
        <ektronUI:TextField ID="uxFriendId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Invite"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            long friendId = long.Parse(uxFriendId.Text);

            FriendsManager friendsManager = new FriendsManager();

            var invitationRequestData = new InvitationSendRequestData();
            invitationRequestData.UserIds.Add(friendId);

            friendsManager.Invite(invitationRequestData);

            MessageUtilities.UpdateMessage(uxMessage, "Invite sent. ",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IsFriend

8.60 and higher

```
IsFriend(System.Int64)
```

Check the user is friend to the currently logged in user.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Friend ID

Parameters

- `friendId`. Friend ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendIdLabel" AssociatedControlID="uxFriendId"
CssClass="span-4 last" runat="server" Text="* Friend Id:" />
        <ektronUI:TextField ID="uxFriendId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Invite"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long friendId = long.Parse(uxFriendId.Text);

            FriendsManager friendsManager = new FriendsManager();
            if (friendsManager.IsFriend(long.Parse(uxFriendId.Text)))
            {
                MessageUtilities.UpdateMessage(uxMessage, uxFriendId.Text +" is friend
to "+new Ektron.Cms.ContentAPI().UserId , Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, uxFriendId.Text +" is not
friend", Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.FriendsData)
```

Updates an existing Friends item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Friend ID
- * Folder ID

Parameters

- `friendsData`. The FriendsData object to update.

Returns

Returns the custom FriendsData object that was updated.

Remarks

Validate the Friends ID and Folder ID with `GetItem()` before you update the properties. Then, call `Update` with your modified FriendsData object.

NOTE: You cannot change all properties after the initial Add event, such as the FriendsData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendIdLabel" AssociatedControlID="uxFriendId"
CssClass="span-4 last" runat="server" Text="* Friend Id:" />
        <ektronUI:TextField ID="uxFriendId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdlabel" AssociatedControlID="uxFolderId"
CssClass="span-4 last" runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long friendId = long.Parse(uxFriendId.Text);

            FriendsManager friendsManager = new FriendsManager();
            FriendsData friendsData = friendsManager.GetItem(friendId);

            if (friendsData != null)
            {
                friendsData.FriendUserId = friendId;
                friendsData.TaxonomyId = (uxFolderId.Text != string.Empty ? long.Parse
(uxFolderId.Text) : friendsData.TaxonomyId);
                friendsManager.Update(friendsData);
                MessageUtilities.UpdateMessage(uxMessage, "Friend updated",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Friend does not exists. ",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

FriendsCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Community
```

Constructors

- `FriendsCriteria()`

```
public FriendsCriteria()
```

- `FriendsCriteria(Ektron.Cms.Common.FriendsProperty, EkEnumeration.OrderByDirection)`

```
public FriendsCriteria(Ektron.Cms.Common.FriendsProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `FriendsProperty` are:

- FriendsUserId
- TaxonomyId
- TaxonomyName
- TaxonomyPath
- UserId

FriendsData

Namespace

```
Ektron.Cms.Community
```

Properties

- `Avatar`

```
public string Avatar { set; get; }
```

- `DisplayName`

```
public string DisplayName { set; get; }
```

- `Email`

```
public string Email { set; get; }
```

- `FirstName`

```
public string FirstName { set; get; }
```

- `FriendsUserId`. Gets or sets the UserId of the friend being added.

```
public long FriendUserId { set; get; }
```

- `LastName`

```
public string LastName { set; get; }
```

- `TaxonomyId`. ID of the Taxonomy to put the friend in for User ID. If not set, it will go in the users root Friends taxonomy folder.

```
public long TaxonomyId { set; get; }
```

- `UserId`. Gets or sets the ID of the user who is adding the friend.

```
public long UserId { set; get; }
```

- `Username`

```
public string Username { set; get; }
```

InvitationSendRequestData

Namespace

```
Ektron.Cms.Common
```

Properties

- `EmailAddress`

```
public System.Collections.ObjectModel.Collection<string>
   EmailAddresses { get; }
```

- `InvitationSendRequestData()`

```
public InvitationSendRequestData()
```

- `NonuserMessageId`

```
public long NonuserMessageId { set; get; }
```

- `OptionalText`

```
public string OptionalText { set; get; }
```

- `SenderId`

```
public long SenderId { set; get; }
```

- `UserIds`. Gets or sets the ID of the user who is adding the friend.

```
public System.Collections.ObjectModel.Collection<long>
  UserIds { get; }
```

- `UserMessageId`. Gets or sets the UserId of the friend being added.

```
public long UserMessageId { set; get; }
```

PendingFriendsData

Namespace

```
Ektron.Cms.Community
```

Properties

- `InvitationId`

```
public long InvitationId { set; get; }
```

FriendsTaxonomyManager

8.50 and higher

The FriendsTaxonomyManager class manages taxonomies of colleague connections (friends).

Namespace

```
Ektron.Cms.Framework.Community.FriendsTaxonomyManager
```

Constructors

- `FriendsTaxonomyManager()`
- `FriendsTaxonomyManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `FriendsFolderManagerService`. Returns an instance of the business objects task manager service.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.FriendTaxonomyData)
```

Adds a friends taxonomy data object.

When you add the friends taxonomy data, the method returns a friends object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- * Parent ID (Taxonomy ID)
- Description

Parameters

- `friendFolderData`. The FriendTaxonomyData object to add.

Returns

Returns the custom FriendTaxonomyData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTitlelabel" AssociatedControlID="uxTitle" CssClass="span-4
last" runat="server" Text="* Title:" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="Friends"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxParentIdLabel" AssociatedControlID="uxParentId"
CssClass="span-4 last" runat="server" Text="* Parent Id(Taxonomy Id):" />
        <ektronUI:TextField ID="uxParentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"/>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last" runat="server" Text=" Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxDescription" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            FriendsTaxonomyManager friendsTaxonomyManager = new FriendsTaxonomyManager
();
            FriendTaxonomyData parentTaxonomy = friendsTaxonomyManager.GetItem
(long.Parse(uxParentId.Text));

            if (parentTaxonomy != null)
            {
                FriendTaxonomyData FriendTaxonomyData = new FriendTaxonomyData()
                {
                    Name = uxTitle.Text,
                    ParentId = parentTaxonomy.ParentId,
                    Description = uxDescription.Text
                };

                friendsTaxonomyManager.Add(FriendTaxonomyData);

                MessageUtilities.UpdateMessage(uxMessage, "Friend Taxonomy Added with Id
" + FriendTaxonomyData.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Parent taxonomy does not
exists. Please check the parent ID.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a friends taxonomy from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `Id`. The identifier of the FriendTaxonomyData item to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendTaxonomyIdLabel"
AssociatedControlID="uxFriendTaxonomyId" CssClass="span-3 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField ID="uxFriendTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long friendTaxonomyId = long.Parse(uxFriendTaxonomyId.Text);

            FriendsTaxonomyManager friendsTaxonomyManager = new FriendsTaxonomyManager
();
            FriendTaxonomyData friendTaxonomyData = friendsTaxonomyManager.GetItem
(friendTaxonomyId);

            if (friendTaxonomyData != null)
            {
                friendsTaxonomyManager.Delete(friendTaxonomyId);
                MessageUtilities.UpdateMessage(uxMessage, "Friend Taxonomy deleted.",
Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Friend
Taxonomy Id.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific FriendTaxonomyData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `Id`. The ID of the FriendTaxonomyData item to get. Internally the folder is stored as Taxonomy so this is the corresponding taxonomy ID.

Returns

FriendTaxonomyData details in a FriendTaxonomyData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendTaxonomyIdLabel"
AssociatedControlID="uxFriendTaxonomyId" CssClass="span-3 last" runat="server" Text="*
Id :" />
        <ektronUI:TextField ID="uxFriendTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxParentId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            long friendTaxonomyId = long.Parse(uxFriendTaxonomyId.Text);

            FriendsTaxonomyManager friendsTaxonomyManager = new FriendsTaxonomyManager
();
            FriendTaxonomyData friendTaxonomyData = friendsTaxonomyManager.GetItem
(friendTaxonomyId);

            if (friendTaxonomyData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for friend taxonomy
with ID " + friendTaxonomyData.Id.ToString() + " are below",
Message.DisplayModes.Success);
                uxName.Text = "Name : " + friendTaxonomyData.Name;
                uxParentId.Text = "Parent Id : " + friendTaxonomyData.ParentId;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Friend taxonomy does not
exist. ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTree

```
GetTree(System.Int64,System.Int32,System.Boolean,Ektron.Cms.PagingInfo)
```

Retrieves the taxonomy tree of a friend.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Depth
- * Get Items

Parameters

- `id`. Taxonomy ID. Internally the folder is stored as Taxonomy so this is the corresponding taxonomy ID.
- `depth`. Depth
- `includeItems`. Should items be included?
- `pagingInfo`. Set paging details.

Returns

FriendTaxonomyData details in a FriendTaxonomyData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendTaxonomyIdLabel"
AssociatedControlID="uxFriendTaxonomyId" CssClass="span-3 last" runat="server" Text="*
Id :" />
        <ektronUI:TextField ID="uxFriendTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDepthLabel" AssociatedControlID="uxDepth" CssClass="span-3
last" runat="server" Text="* Depth :" />
        <ektronUI:TextField ID="uxDepth" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxGetItemsLabel" AssociatedControlID="uxGetItems"
CssClass="span-3 last" runat="server" Text="* Get Items:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxGetItems" CssClass="span-
6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxParentId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxItems" runat="server"></asp:Literal>
```

```
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            FriendsTaxonomyManager friendsTaxonomyManager = new FriendsTaxonomyManager
();
            FriendTaxonomyData friendTaxonomyData = friendsTaxonomyManager.GetTree
(long.Parse(uxFriendTaxonomyId.Text), int.Parse(uxDepth.Text), uxGetItems.Checked, new
PagingInfo());

            if (friendTaxonomyData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for friend Taxonomy
with ID " + friendTaxonomyData.Id.ToString() + " are below",
Message.DisplayModes.Success);
                uxName.Text = "Name : " + friendTaxonomyData.Name;
                uxParentId.Text = "Parent Id : " + friendTaxonomyData.ParentId;
                uxItems.Text = "Item Count :" + friendTaxonomyData.Items.Count.ToString
() ;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Friend Taxonomy does not
exist. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetUserTaxonomy

```
GetUserTaxonomy(System.Int64)
```

Retrieves the taxonomy tree of a friend.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `userId`. The user ID of the user for which to retrieve the friends taxonomy.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendTaxonomyIdLabel"
AssociatedControlID="uxFriendTaxonomyId" CssClass="span-3 last" runat="server"
Text="*User Id :" />
        <ektronUI:TextField ID="uxFriendTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxParentId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Framework.User;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long userId = long.Parse(uxFriendTaxonomyId.Text);

            UserManager Usermanager = new UserManager(ApiAccessMode.Admin);
            //Check Whether entered UserId valid or Not
            UserData Userdata = Usermanager.GetItem(userId);
            if (Userdata != null)
            {
                FriendsTaxonomyManager friendsTaxonomyManager = new
FriendsTaxonomyManager();
                FriendTaxonomyData friendTaxonomyData =
friendsTaxonomyManager.GetUserTaxonomy(userId);

                if (friendTaxonomyData != null)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Details for friend
Taxonomy with ID " + friendTaxonomyData.Id.ToString() + " are below",
Message.DisplayModes.Success);
                    uxName.Text = "Name : " + friendTaxonomyData.Name;
                    uxParentId.Text = "Parent Id : " + friendTaxonomyData.ParentId;
                    uxFriendsCount.Text = "Friends : " + friendTaxonomyData.Items.Count;
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Friend User Taxonomy does
not exist.", Message.DisplayModes.Error);
                    return;
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User ID and
Try again.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.FriendTaxonomyData)
```

Updates an existing friends taxonomy item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Friend Taxonomy ID
- * Title
- Description

Parameters

- `friendTaxonomyData`. The FriendTaxonomyData object to update.

Returns

Returns the custom FriendTaxonomyData object updated.

Remarks

Validate the FriendsTaxonomy item with `GetItem()` before you update the properties. Then, call `Update` with your modified FriendsTaxonomyData object.

NOTE: You cannot change all properties after the initial Add event, such as the FriendsTaxonomyData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFriendTaxonomyIdLabel"
AssociatedControlID="uxFriendTaxonomyId" CssClass="span-4 last" runat="server" Text="*
Friend Taxonomy Id:" />
        <ektronUI:TextField ID="uxFriendTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTitlelabel" AssociatedControlID="uxTitle" CssClass="span-4
last" runat="server" Text="* Title:" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last" runat="server" Text=" Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxDescription" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long friendTaxonomyId = long.Parse(uxFriendTaxonomyId.Text);

            FriendsTaxonomyManager friendsTaxonomyManager = new FriendsTaxonomyManager
();
            FriendTaxonomyData friendTaxonomyData = friendsTaxonomyManager.GetItem
(friendTaxonomyId);

            if (friendTaxonomyData != null)
            {
                friendTaxonomyData.Name = uxTitle.Text != string.Empty ? uxTitle.Text :
friendTaxonomyData.Name;
                friendTaxonomyData.Description = uxDescription.Text != string.Empty ?
uxDescription.Text : friendTaxonomyData.Description;

                friendsTaxonomyManager.Update(friendTaxonomyData);
                MessageUtilities.UpdateMessage(uxMessage,  "Friend Taxonomy updated",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Friend Taxonomy does not
exist. ", Message.DisplayModes.Error);
```

```
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

FriendTaxonomyData

Namespace

```
Ektron.Cms.Community
```

Properties

- `ChildTaxonomies`. Gets or sets the Friends child taxonomy in the current taxonomy. If not retrieved, this will be an empty list.

  ```
  public System.Collections.Generic.List<FriendTaxonomyData>
    ChildTaxonomies { set; get; }
  ```

- `Description`. Gets or sets the description of the taxonomy's folder.

  ```
  public string Description { set; get; }
  ```

- `FriendTaxonomyData()`

  ```
  public FriendTaxonomyData()
  ```

- `Id`. Gets or sets the ID of the Friends taxonomy. Overridden to clarify the description and taxonomy relationship

  ```
  public override long Id { set; get; }
  ```

- `Items`. Gets or sets the Friends in the taxonomy. If not retrieved, this will be an empty list.

  ```
  public System.Collections.Generic.List<FriendsData> Items { set; get; }
  ```

- `Name`. Gets or sets the Friends taxonomy name.

  ```
  public string Name { set; get; }
  ```

- `ParentId`. Gets or sets the ID of the Friends folder's parent taxonomy.

  ```
  public long ParentId { set; get; }
  ```

MessageBoardManager

8.50 and higher

The MessageBoardManager class manages items that are posted on a message board.

Namespace

```
Ektron.Cms.Framework.Community.MessageBoardManager
```

Constructors

- `MessageBoardManager()`
- `MessageBoardManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `MessageBoardManagerService`. Returns an instance of the business objects messageboard manager service.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- `IsUserMessageBoardAdmin(Int64,MessageBoardObjectType,Int64)`. 8.60 and higher Checks whether a user is a MessageBoard admin.
- *Update* on page 973

Add

```
Add(Ektron.Cms.MessageBoardData)
```

Adds a message board post with details from the supplied MessageBoardData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object Type
- * Object Value
- Message Text

Parameters

- `messageBoardData`. Message board post data to add.

Returns

Returns the custom MessageBoardData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageBoardObjectTypeLabel"
AssociatedControlID="uxMessageBoardObjectType" CssClass="span-4 last" runat="server"
Text="* Object Type:" />
        <asp:DropDownList ID="uxMessageBoardObjectType" runat="server">
            <asp:ListItem>Community Group</asp:ListItem>
            <asp:ListItem>User</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageTextLabel" AssociatedControlID="uxMessageText"
CssClass="span-4 last" runat="server" Text="* Message Text:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxMessageText" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            MessageBoardManager messageBoardManager = new MessageBoardManager();

            CommunityGroupManager communityGroupManager = new CommunityGroupManager();
            CommunityGroupData communityGroupData =
(uxMessageBoardObjectType.SelectedItem.Text == "Community Group" ?
communityGroupManager.GetItem(Convert.ToInt64(uxObjectValue.Text)) : null);

            if (communityGroupData != null || uxMessageBoardObjectType.SelectedItem.Text
!= "Community Group")
            {
                MessageBoardData messageBoardData = new MessageBoardData()
                {
                    ObjectType = (uxMessageBoardObjectType.SelectedItem.Text ==
"Community Group" ?
Ektron.Cms.Common.EkEnumeration.MessageBoardObjectType.CommunityGroup :
Ektron.Cms.Common.EkEnumeration.MessageBoardObjectType.User),
                    ObjectId = long.Parse(uxObjectValue.Text),
                    MessageText = uxMessageText.Text,
                    UserId = messageBoardManager.RequestInformation.UserId
                };

                messageBoardManager.Add(messageBoardData);
                MessageUtilities.UpdateMessage(uxMessage, "Message Board message added
with Id " + messageBoardData.Id, Message.DisplayModes.Success);
            }
            else
            {
```

```
                MessageUtilities.UpdateMessage(uxMessage, "Community Group Object Value
is invalid.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Approve (post)

```
Approve(System.Int64)
```

Approves a message board post.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the message board post.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageBoardIdLabel"
AssociatedControlID="uxMessageBoardId" CssClass="span-3 last" runat="server" Text="* Id
:" />
        <ektronUI:TextField ID="uxMessageBoardId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Approve"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageBoardId = long.Parse(uxMessageBoardId.Text);

            MessageBoardManager messageBoardManager = new MessageBoardManager();
            messageBoardManager.Approve(messageBoardId);

            MessageUtilities.UpdateMessage(uxMessage, "Message board message approved.",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Approve (list)

```
Approve(System.Collections.Generic.List)
```

Approves a list of messages to the message board.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * IDs (comma separated)

Parameters

- `ids`. The list of the IDs.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageBoardIdLabel"
AssociatedControlID="uxMessageBoardId" CssClass="span-4 last" runat="server" Text="* Ids
(Comma Seperated) :" />
        <ektronUI:TextField ID="uxMessageBoardId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Approve"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            List<long> listMessageBoardIds = new List<long>();
            string messageBoardIds = uxMessageBoardId.Text;
            string[] strMessageBoardIds = messageBoardIds.Split(new char[] { ',' });

            foreach (string str in strMessageBoardIds)
            {
                listMessageBoardIds.Add(Convert.ToInt64(str));
            }

            MessageBoardManager messageBoardManager = new MessageBoardManager();
            messageBoardManager.Approve(listMessageBoardIds);

            MessageUtilities.UpdateMessage(uxMessage, "Message Boards with Ids " +
messageBoardIds + " approved.", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete (post)

```
Delete(System.Int64)
```

Deletes a message board post from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the message board post to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageBoardIdLabel"
AssociatedControlID="uxMessageBoardId" CssClass="span-3 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField ID="uxMessageBoardId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                    <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                        Text="Status:" />
                </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long messageBoardId = long.Parse(uxMessageBoardId.Text);

            MessageBoardManager messageBoardManager = new MessageBoardManager();
            MessageBoardData messageBoardData = messageBoardManager.GetItem
(messageBoardId);

            if (messageBoardData != null && messageBoardData.Id > 0)
            {
                messageBoardManager.Delete(messageBoardId);
                MessageUtilities.UpdateMessage(uxMessage,  "Message Board with Object Id
" + messageBoardId + " deleted.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter valid
messageBoardId.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete (list)

```
Delete(System.Collections.Generic.List)
```

Deletes a list of message board posts.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object ID
- * Object Type

Parameters

- `ids`. A comma-separated list of the IDs.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageBoardIdLabel"
AssociatedControlID="uxMessageBoardId" CssClass="span-6 last" runat="server" Text="* ids
(Comma Seperated):" />
        <ektronUI:TextField ID="uxMessageBoardId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            List<long> listMessageBoardIds = new List<long>();
            string messageBoardIds = uxMessageBoardId.Text;
            string[] strMessageBoardIds = messageBoardIds.Split(new char[] { ',' });
```

```
            foreach (string str in strMessageBoardIds)
            {
                listMessageBoardIds.Add(Convert.ToInt64(str));
            }

            MessageBoardManager messageBoardManager = new MessageBoardManager();
            messageBoardManager.Delete(listMessageBoardIds);

            MessageUtilities.UpdateMessage(uxMessage, "Message Boards with Ids " +
messageBoardIds + "  deleted.", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete (object)

```
Delete(System.Int64,Ektron.Cms.Common.EkEnumeration.MessageBoardObjectType)
```

Deletes all message board posts of a specified object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object ID
- * Object Type

Parameters

- `objectId`. ID of the object for which to delete posts. The type of object identified by the ID is defined by the MessageBoardObjectType parameter.
- `objectType`. The type of the object for which to delete posts.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageBoardIdLabel"
AssociatedControlID="uxMessageBoardId" CssClass="span-3 last" runat="server" Text="*
```

```
Object Id:" />
        <ektronUI:TextField ID="uxMessageBoardId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
            <ektronUI:Label ID="uxMessageBoardObjectTypeLabel"
AssociatedControlID="uxMessageBoardObjectType" CssClass="span-3 last" runat="server"
Text="* Object Type:" />
            <asp:DropDownList ID="uxMessageBoardObjectType" runat="server">
                <asp:ListItem>Community Group</asp:ListItem>
                <asp:ListItem>User</asp:ListItem>
            </asp:DropDownList>
        </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long objectId = long.Parse(uxMessageBoardId.Text);
            EkEnumeration.MessageBoardObjectType ObjectType =
(uxMessageBoardObjectType.SelectedItem.Text == "Community Group" ?
EkEnumeration.MessageBoardObjectType.CommunityGroup :
EkEnumeration.MessageBoardObjectType.User);

            MessageBoardManager messageBoardManager = new MessageBoardManager();
            messageBoardManager.Delete(objectId, ObjectType);

            MessageUtilities.UpdateMessage(uxMessage, "Message Board with Object Id " +
objectId + " and Object Type " + ObjectType.ToString() + "  deleted.",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
```

```
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific MessageBoardData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the message board to get.

Returns

Message board details in a MessageBoardData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageBoardIdLabel"
AssociatedControlID="uxMessageBoardId" CssClass="span-3 last" runat="server" Text="* Id
:" />
        <ektronUI:TextField ID="uxMessageBoardId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
```

```
        <asp:Literal ID="uxObjectId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxObjectType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxMessageText" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsModerated" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageBoardId = long.Parse(uxMessageBoardId.Text);

            MessageBoardManager messageBoardManager = new MessageBoardManager();
            MessageBoardData messageBoardData = messageBoardManager.GetItem
(messageBoardId);

            if (messageBoardData != null && messageBoardData.Id > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for message board
with ID " + messageBoardId + " are below", Message.DisplayModes.Success);
                uxObjectId.Text = "Object Id : " + messageBoardData.ObjectId;
                uxObjectType.Text = "Object Type : " +
messageBoardData.ObjectType.ToString();
                uxMessageText.Text = "Message Text : " + messageBoardData.MessageText;
                uxUserName.Text = "User Name : " + messageBoardData.UserName;
                uxIsModerated.Text = "Is Moderated :" + messageBoardData.IsModerated;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Message board does not
exists. ", Message.DisplayModes.Error);
            }
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(MessageBoardCriteria)
```

Retrieves lists of objects through the GetList(MessageBoardCriteria) method.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Community Group Property
- * Object Value

Parameters

- `criteria`. Used to specify, or filter, the message board to return.

Returns

A list of message boards.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageBoardPropertyLabel"
AssociatedControlID="uxMessageBoardProperty" CssClass="span-6 last" runat="server"
Text="CommunityGroupProperty:" />
        <asp:DropDownList ID="uxMessageBoardProperty" runat="server">
            <asp:ListItem>Object Id</asp:ListItem>
            <asp:ListItem>Object Type (0,1,2,3)</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxMessageBoardListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Object Id
                    </th>
                    <th>
                        Object Type
                    </th>
                    <th>
                        Is Moderated
                    </th>
                    <th>
                        Message Text
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
                <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectType")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsModerated")%>
            </td>
            <td class="devsite-method">
                <%# Eval("MessageText")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            Ektron.Cms.Community.MessageBoardCriteria criteria = new
Ektron.Cms.Community.MessageBoardCriteria();
            if (uxMessageBoardProperty.SelectedItem.Text == "Object Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(MessageboardProperty.ObjectId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(MessageboardProperty.ObjectType,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            MessageBoardManager messageBoardManager = new MessageBoardManager();
            List<MessageBoardData> messageBoardDataList = messageBoardManager.GetList
(criteria);

            uxMessageBoardListView.Visible = true;
            uxMessageBoardListView.DataSource = messageBoardDataList;
            uxMessageBoardListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetReplyList

```
GetReplyList(System.Int64)
```

Retrieves a list of replies for a specified message.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the message.

Returns

A list of message boards.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageBoardIdLabel"
AssociatedControlID="uxMessageBoardId" CssClass="span-6 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField ID="uxMessageBoardId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Replies"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxMessageBoardListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Object Id
                    </th>
                    <th>
                        Object Type
                    </th>
```

```
                    <th>
                        Message Text
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
                <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectType")%>
            </td>
            <td class="devsite-method">
                <%# Eval("MessageText")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageBoardId = long.Parse(uxMessageBoardId.Text);

            MessageBoardManager messageBoardManager = new MessageBoardManager();
            List<MessageBoardData> messageBoardData = messageBoardManager.GetReplyList
(messageBoardId);

            uxMessageBoardListView.Visible = true;
            uxMessageBoardListView.DataSource = messageBoardData;
            uxMessageBoardListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
```

```
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetUnapprovedList

```
GetUnapprovedList(MessageBoardObjectType,Int64,Paginginfo)
```

Retrieves a list of unapproved messages from the message board for a specified message.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object Type
- * Object ID
- * Records per page

Parameters

- `ObjectType`. The type of the object.
- `ObjectID`. The ID of the object.
- `RecordsPerPage`. The number of records per page.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageBoardPropertyLabel"
AssociatedControlID="uxObjectType" CssClass="span-4 last" runat="server" Text="* Object
Type:" />
        <asp:DropDownList ID="uxObjectType" runat="server">
            <asp:ListItem>Content</asp:ListItem>
            <asp:ListItem>User</asp:ListItem>
            <asp:ListItem>CommunityGroup</asp:ListItem>
            <asp:ListItem>Messageboard Reply</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxObjectId"
CssClass="span-4 last" runat="server" Text="* Object Id:" />
        <ektronUI:TextField ID="uxObjectId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxRecordsPPLabel1" AssociatedControlID="uxRecordsPP"
CssClass="span-4 last" runat="server" Text="* Records per Page:" />
        <ektronUI:TextField ID="uxRecordsPP" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="100" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxMessageBoardListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Object Id
                    </th>
                    <th>
                        Object Type
                    </th>
                    <th>
                        Is Moderated
                    </th>
                    <th>
                        Message Text
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
                <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectType")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsModerated")%>
            </td>
            <td class="devsite-method">
                <%# Eval("MessageText")%>
```

```
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long objectId = long.Parse(uxObjectId.Text);
            EkEnumeration.MessageBoardObjectType objectType =
(EkEnumeration.MessageBoardObjectType)uxObjectType.SelectedIndex;

            PagingInfo pagingInfo = new PagingInfo();
            pagingInfo.RecordsPerPage = int.Parse(uxRecordsPP.Text);

            MessageBoardManager messageBoardManager = new MessageBoardManager();
            List<MessageBoardData> messageBoardDataList =
messageBoardManager.GetUnapprovedList(objectType, objectId, pagingInfo);

            uxMessageBoardListView.Visible = true;
            uxMessageBoardListView.DataSource = messageBoardDataList;
            uxMessageBoardListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.FriendTaxonomyData)
```

Updates an existing message board item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Message ID
- * Message Text

Parameters

- `messageBoardData`. Message board post data to update.

Returns

Returns the custom MessageBoardData object updated.

Remarks

Validate the message board item with `GetItem()` before you update the properties. Then, call `Update` with your modified MessageBoardData object.

NOTE: You cannot change all properties after the initial Add event, such as the MessageBoardData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* Message Id:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageTextLabel" AssociatedControlID="uxMessageText"
CssClass="span-4 last" runat="server" Text="* Message Text:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxMessageText" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            MessageBoardManager messageBoardManager = new MessageBoardManager();

            long objectId = long.Parse(uxObjectValue.Text);

            MessageBoardData messageBoardData = messageBoardManager.GetItem(objectId);
            if (messageBoardData != null)
            {
                messageBoardData.MessageText = uxMessageText.Text != string.Empty ?
uxMessageText.Text : messageBoardData.MessageText;
                messageBoardManager.Update(messageBoardData);

                MessageUtilities.UpdateMessage(uxMessage,  "Message Board message
updated. ", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Message Board does not
exists.  ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

MessageBoardCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Community
```

Constructors

- `MessageBoardCriteria()`

```
public MessageBoardCriteria()
```

- `MessageBoardCriteria(Ektron.Cms.Common.MessageBoardProperty, EkEnumeration.OrderByDirection)`

```
public MessageBoardCriteria(Ektron.Cms.Common.MessageBoardProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `MessageBoardProperty` are:

- DateCreated
- DateModified
- DisplayName
- HasPendingReplies
- HasReplies
- IsModerated
- LanguageId
- MessageId
- ObjectId
- ObjectType
- ReplayCount
- Status
- UserId
- UserName

MessageBoardData

Namespace

```
Ektron.Cms
```

Properties

- `AllowApproval`

```
public bool AllowApproval { set; get; }
```

- `AllowDelete`

```
public bool AllowDelete { set; get; }
```

- `AllowModeration`

```
public bool AllowModeration { set; get; }
```

- `DateCreated`

```
public System.DateTime DateCreated { set; get; }
```

- `DateModified`

```
public System.DateTime DateModified { set; get; }
```

- DisplayName

```
public string DisplayName { set; get; }
```

- DisplayUserName

```
public string DisplayUserName { set; get; }
```

- HasPendingReplies

```
public bool HasPendingReplies { get; }
```

- HasReplies

```
public bool HasReplies { get; }
```

- Id

```
public override long Id { set; get; }
```

- IsModerated. Gets or sets the IsModerated flag. This flag is used on ADD to determine if the message board post should follow moderation. On retrieval, it will not be set because the message board post itself has no concept of moderation.

```
public bool IsModerated { set; get; }
```

- LanguageId

```
public int LanguageId { set; get; }
```

- MessageId

```
public long MessageId { set; get; }
```

- MessageText

```
public string MessageText { set; get; }
```

- ObjectId

```
public long ObjectId { set; get; }
```

- ObjectType

```
public EkEnumeration.MessageBoardObjectType ObjectType { set; get; }
```

- PendingReplies

```
public int PendingReplies { set; get; }
```

- ReplyCount

```
public int ReplyCount { set; get; }
```

- SiteId

```
public long SiteId { set; get; }
```

- Status

```
public string Status { set; get; }
```

- UserAvatar

```
public string UserAvatar { set; get; }
```

- UserEmailAddress

```
public string UserEmailAddress { set; get; }
```

- UserFirstname

```
public string UserFirstName { set; get; }
```

- UserId

```
public long UserId { set; get; }
```

- UserLastName

```
public string UserLastname { set; get; }
```

- UserName

```
public string UserName { set; get; }
```

- UserRole

```
public string UserRole { set; get; }
```

MessageBoardObjectType

Namespace

```
Ektron.Cms.Common.EkEnumeration
```

Properties

- CommunityGroup

```
public Const CommunityGroup As
  Ektron.Cms.Common.EkEnumeration.MessageBoardObjectType = 2
```

- Content

```
public Const Content As
  Ektron.Cms.Common.EkEnumeration.MessageBoardObjectType = 0
```

- MessageReply

```
public Const MessageReply As
  Ektron.Cms.Common.EkEnumeration.MessageBoardObjectType = 3
```

- User

```
public Const User As
  Ektron.Cms.Common.EkEnumeration.MessageBoardObjectType = 1
```

MicromessageManager

8.50 and higher

The MicromessageManager class manages user status messages that are posted to an activity stream.

Namespace

```
Ektron.Cms.Framework.Community.MicromessageManager
```

Constructors

- `MicromessageManager()`
- `MicromessageManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `MicromessageManagerService`. Returns an instance of the business objects micromessage manager service.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *GetUserMessageList* on page 997
- *IsFullTextSearchInstalled* on page 1000
- *IsSpam* on page 1001

Add

```
Add(Ektron.Cms.MicroMessageData)
```

Adds a micro-message with details from the supplied MicroMessageData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Message Text

Parameters

- `data`. The micro-message data.

Returns

Returns the custom MicroMessageData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageTextLabel" AssociatedControlID="uxMessageText"
CssClass="span-4 last" runat="server" Text="* Message Text:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxMessageText" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MicromessageManager microMessageManager = new MicromessageManager();
            MicroMessageData microMessageData = new MicroMessageData()
            {
                MessageText = uxMessageText.Text ,
                SiteId = 0,
                UserId = microMessageManager.RequestInformation.UserId
            };

            microMessageManager.Add(microMessageData);

            MessageUtilities.UpdateMessage(uxMessage, "Micro message added with Id " +
microMessageData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

AddReply

```
AddReply(System.Int64,System.String)
```

Adds a new reply.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Message ID
- * Reply Text

Parameters

- `parentMessageId`. ID of the micro-message to which to add a reply.
- `messageText`. The text of the reply.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageIdLabel" AssociatedControlID="uxMessageId"
CssClass="span-4 last" runat="server" Text="* Message Id:" />
        <ektronUI:TextField ID="uxMessageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageTextLabel" AssociatedControlID="uxMessageText"
CssClass="span-4 last" runat="server" Text="* Reply Text:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxMessageText" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Add
Reply"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageId = long.Parse(uxMessageId.Text);

            MicromessageManager micromessageManager = new MicromessageManager();
            micromessageManager.AddReply(messageId, uxMessageText.Text);

            MessageUtilities.UpdateMessage(uxMessage, "Reply added ",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a micro-message from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `Id`. ID of the micro-message to be deleted.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageIdLabel" AssociatedControlID="uxMessageId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxMessageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms.Framework.Community;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageBoardId = long.Parse(uxMessageId.Text);

            MicromessageManager microMessageManager = new MicromessageManager();
            MicroMessageData microMessageData = microMessageManager.GetItem
(messageBoardId);
            if (microMessageData != null)
            {
                microMessageManager.Delete(messageBoardId);
                MessageUtilities.UpdateMessage(uxMessage, "Micro Message deleted.",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Micro Message doesn't
exists.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetColleagueMessageList

```
GetColleagueMessageList(Ektron.Cms.PagingInfo)
```

Retrieves a list of colleague messages.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Records per page

Parameters

- `paging`. The paging object. Setting `RecordsPerPage = 0` retrieves all rows.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRecordsLabel" AssociatedControlID="uxRecords"
CssClass="span-4 last" runat="server" Text="* Records per page:" />
        <ektronUI:TextField ID="uxRecords" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="10" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxMicroMessageDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Id
                    </th>
                    <th>
                        Message Text
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
                <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("MessageText")%>
            </td>
        </tr>
```

```
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PagingInfo pagingInfo = new PagingInfo();
            pagingInfo.RecordsPerPage = int.Parse(uxRecords.Text);

            MicromessageManager micromessageManager = new MicromessageManager();
            List<MicroMessageData> microMessageDataList =
micromessageManager.GetColleagueMessageList(pagingInfo);

            uxMicroMessageDataListView.Visible = true;
            uxMicroMessageDataListView.DataSource = microMessageDataList;
            uxMicroMessageDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific MicroMessageData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the micro-message.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageIdLabel" AssociatedControlID="uxMessageId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxMessageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxMessageText" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxReplyCount" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            long messageId = long.Parse(uxMessageId.Text);

            MicromessageManager micromessageManager = new MicromessageManager();
            MicroMessageData microMessageData = micromessageManager.GetItem(messageId);

            if (microMessageData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for message with ID "
+ messageId + " are below", Message.DisplayModes.Success);
                uxUserId.Text = "User Id : " + microMessageData.UserId;
                uxMessageText.Text = "Message Text : " + microMessageData.MessageText;
                uxUserName.Text = "User Name : " + microMessageData.UserName;
                uxReplyCount.Text = "Reply Count :" + microMessageData.ReplyCount;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Message does not exists. ",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(MicromessageCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Micro-message Property
- * Object Value

Parameters

- `criteria`. MicroMessageCriteria used to filter results.

See *Criteria use for GetList methods* on page 156 for additional information.

Remarks

Use ListView because it has a paging option and data grouping.

Use the default page size = 50.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMicroMessagePropertyLabel"
AssociatedControlID="uxMicroMessageProperty" CssClass="span-6 last" runat="server"
Text="* MicroMessageProperty:" />
        <asp:DropDownList ID="uxMicroMessageProperty" runat="server">
            <asp:ListItem>UserId</asp:ListItem>
            <asp:ListItem>MessageText</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxMicroMessageDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Id
                    </th>
                    <th>
                        Message Text
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
```

```
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
                <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("MessageText")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            MicroMessageCriteria criteria = new MicroMessageCriteria
(MicroMessageProperty.MessageId, EkEnumeration.OrderByDirection.Ascending);
            if (uxMicroMessageProperty.SelectedItem.Text == "UserId")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(MicroMessageProperty.UserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(MicroMessageProperty.MessageText,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            MicromessageManager micromessageManager = new MicromessageManager();
            List<MicroMessageData> microMessageDataList = micromessageManager.GetList
(criteria);
```

```
            uxMicroMessageDataListView.Visible = true;
            uxMicroMessageDataListView.DataSource = microMessageDataList;
            uxMicroMessageDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetPublicMessageList

```
GetPublicMessageList(Ektron.Cms.PagingInfo)
```

Retrieves a list of public and colleague messages that the requesting user has permission to see.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Records per page

Parameters

- `paging`. The paging object. Setting `RecordsPerPage = 0` retrieves all rows.

Returns

Returns a list of for the user messages.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRecordsLabel" AssociatedControlID="uxRecords"
CssClass="span-6 last" runat="server" Text="* Records per page:" />
        <ektronUI:TextField ID="uxRecords" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="10" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxMicroMessageDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Id
                    </th>
                    <th>
                        Message Text
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
                <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("MessageText")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PagingInfo pagingInfo = new PagingInfo();
            pagingInfo.RecordsPerPage = int.Parse(uxRecords.Text);

            MicromessageManager micromessageManager = new MicromessageManager();
            List<MicroMessageData> microMessageDataList =
micromessageManager.GetPublicMessageList(pagingInfo);

            uxMicroMessageDataListView.Visible = true;
            uxMicroMessageDataListView.DataSource = microMessageDataList;
            uxMicroMessageDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetReplyList

```
GetReplyList(System.Int64)
```

Retrieves a list of replies for a specified micro-message.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Message ID

Parameters

- `parentMessageId`. ID of the micro-message that has replies.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageIdLabel" AssociatedControlID="uxMessageId"
CssClass="span-4 last" runat="server" Text="* Message Id:" />
        <ektronUI:TextField ID="uxMessageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Reply List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxMicroMessageDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Id
                    </th>
                    <th>
                        Message Text
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
                <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("MessageText")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageId = long.Parse(uxMessageId.Text);

            MicromessageManager micromessageManager = new MicromessageManager();
            List<MicroMessageData> microMessageDataList =
micromessageManager.GetReplyList(messageId);

            uxMicroMessageDataListView.Visible = true;
            uxMicroMessageDataListView.DataSource = microMessageDataList;
            uxMicroMessageDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetSearchList

```
GetSearchList(System.String, System.Int64)
```

Retrieves the search results.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Search Text
- * Records per page

Parameters

- `searchText`. The text string to search.
- `paging`. The paging object. Setting `RecordsPerPage = 0` retrieves all rows.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxSearchTextLabel" AssociatedControlID="uxSearchText"
CssClass="span-4 last" runat="server" Text="* Search Text:" />
        <ektronUI:TextField ID="uxSearchText" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRecordsLabel" AssociatedControlID="uxRecords"
CssClass="span-4 last" runat="server" Text="* Records per page:" />
        <ektronUI:TextField ID="uxRecords" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="10" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxMicroMessageDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Id
                    </th>
                    <th>
                        Message Text
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
                <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
```

```
            </td>
            <td class="devsite-method">
                <%# Eval("MessageText")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PagingInfo pagingInfo = new PagingInfo();
            pagingInfo.RecordsPerPage = int.Parse(uxRecords.Text);

            MicromessageManager micromessageManager = new MicromessageManager();
            List<MicroMessageData> microMessageDataList =
micromessageManager.GetSearchList(uxSearchText.Text, pagingInfo);

            uxMicroMessageDataListView.Visible = true;
            uxMicroMessageDataListView.DataSource = microMessageDataList;
            uxMicroMessageDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetUserMessageList

```
GetUserMessageList(System.Int64,Ektron.Cms.PagingInfo)
```

Retrieves a list of sent user messages that the requesting user has permission to see.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `paging`. The paging object. Setting `RecordsPerPage = 0` retrieves all rows.
- `userId`. ID for which to get message list.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-6 last" runat="server" Text="* User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxMicroMessageDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Id
                    </th>
                    <th>
                        Message Text
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
```

```
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
                <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("MessageText")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long userId = long.Parse(uxUserId.Text);

            PagingInfo pagingInfo = new PagingInfo();
            pagingInfo.RecordsPerPage = 10;

            MicromessageManager micromessageManager = new MicromessageManager();
            List<MicroMessageData> microMessageDataList =
micromessageManager.GetUserMessageList(userId, pagingInfo);

            uxMicroMessageDataListView.Visible = true;
            uxMicroMessageDataListView.DataSource = microMessageDataList;
            uxMicroMessageDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IsFullTextSearchInstalled

```
IsFullTextSearchInstalled
```

Checks whether full text search is installed on the database. This method has no parameters.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

.aspx code snippet

```
<ol class="formFields">
<li class="clearfix">
<ektronUI:Button id="uxSubmit" runat="server"
 OnClick="uxSubmit_Click" Text="IsFullTextSearchInstalled">
</ektronUI:Button>
</li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MicromessageManager micromessageManager = new MicromessageManager();
            bool isFullTextSearchInstalled =
micromessageManager.IsFullTextSearchInstalled();

            MessageUtilities.UpdateMessage(uxMessage, "Is Full Text Search Installed: "
+ isFullTextSearchInstalled.ToString(), Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IsSpam

```
IsSpam
(Ektron.Cms.Common.EkEnumeration.MicroMessageSpamControlType,System.Int32,Ektron.Cms.Mic
roMessageData)
```

Checks whether a micro-message is spam.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Spam Timespan

Parameters

- `spamControlType`. SpamControlType
- `SpamTimeSpan`. SpamTimeSpan
- `data`. Data

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxMessageIdLabel" AssociatedControlID="uxMessageId"
CssClass="span-3 last" runat="server" Text="* Id :" />
            <ektronUI:TextField ID="uxMessageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" text="0"/>
        </li>
        <li class="clearfix">
            <ektronUI:Label ID="uxSpamTimeSpanLabel"
AssociatedControlID="uxSpamTimeSpan" CssClass="span-3 last" runat="server" Text="*
SpamTimeSpan :" />
            <ektronUI:TextField ID="uxSpamTimeSpan" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" text="5"/>
        </li>
        <li class="clearfix">
            <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="IsSpam"></ektronUI:Button>
            <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server"
Text="* - Required" />
        </li>
    </ol>
</div>
</asp:View>
<asp:View ID="uxViewMessage" runat="server">
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageId = long.Parse(uxMessageId.Text);

            MicromessageManager micromessageManager = new MicromessageManager();
            MicroMessageData microMessageData = micromessageManager.GetItem(messageId);

            if (microMessageData != null)
            {
                bool isSpam = micromessageManager.IsSpam
(Ektron.Cms.Common.EkEnumeration.MicroMessageSpamControlType.SameUserTimeDelay,
int.Parse(uxSpamTimeSpan.Text), microMessageData);

                if (isSpam)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Message is IsSpam",
Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Message is Not a IsSpam",
Message.DisplayModes.Success);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Message does not exists. ",
Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

MicroMessageCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms
```

Constructors

- `MicroMessageCriteria()`

  ```
  public MicroMessageCriteria()
  ```

- `MicroMessageCriteria(Ektron.Cms.MicroMessageProperty, EkEnumeration.OrderByDirection)`

  ```
  public MicroMessageCriteria(Ektron.Cms.MicroMessageProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `MicroMessageProperty` are:

  - Avatar
  - DateCreated
  - DateModified
  - DisplayName
  - EmailAddress
  - FirstName
  - LastName
  - MessaseId
  - ParentMessageId
  - ReplyCount
  - SiteId
  - UserId
  - UserName

MicroMessageData

Namespace

```
Ektron.Cms
```

Properties

- `AllowDelete`. Gets or sets the AllowDelete for this micromessage.

  ```
  public bool AllowDelete { set; get; }
  ```

- `DateCreated`. Gets or sets the DateCreated for this micromessage.

```
public System.DateTime DateCreated { set; get; }
```

- `DateModified`. Gets or sets the DateModified for this micromessage.

```
public System.DateTime DateModified { set; get; }
```

- `DisplayName`. Gets or sets the DisplayName for this micromessage.

```
public string DisplayName { set; get; }
```

- `DisplayUserName`. Gets or sets the DisplayUserName for this micromessage.

```
public string DisplayUserName { set; get; }
```

- `HasReplies`. Gets if this micromessage has replies.

```
public bool HasReplies { get; }
```

- `Id`. Gets or sets the Id for this micromessage.

```
public long Id { set; get; }
```

- `IsReply`. Gets if this micromessage is a reply.

```
private bool IsReply { get; }
```

- `MessageText`. Gets or sets the MessageText for this micromessage.

```
public string MessageText { set; get; }
```

- `ParentMessageId`. Gets or sets the ParentMessageId for this micromessage.

```
public long ParentMessageId { set; get; }
```

- `ReplyCount`. Gets or sets the ReplyCount for this micromessage.

```
public int ReplyCount { set; get; }
```

- `SiteId`. Gets or sets the SiteId for this micromessage.

```
public long SiteId { set; get; }
```

- `TimeLapse`. Gets or sets the TimeLapse for this micromessage.

```
public string TimeLapse { set; get; }
```

- `UserAvatar`. Gets or sets the UserAvatar for this micromessage.

```
public string UserAvatar { set; get; }
```

- `UserEmailAddress`. Gets or sets the UserEmailAddress for this micromessage.

```
public string UserEmailAddress { set; get; }
```

- `UserFirstName`. Gets or sets the UserFirstName for this micromessage.

```
public string UserFirstName { set; get; }
```

- `UserId`. Gets or sets the UserId for this micromessage.

```
public long UserId { set; get; }
```

- `UserLastname`. Gets or sets the UserLastName for this micromessage.

```
public string UserLastname { set; get; }
```

- `Username`. Gets or sets the UserName for this micromessage.

```
public string UserName { set; get; }
```

PrivateMessageManager

8.50 and higher

The PrivateMessageManager class manages messages between users.

Namespace

```
Ektron.Cms.Framework.Community.PrivateMessageManager
```

Constructors

- `PrivateMessageManager()`
- `PrivateMessageManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `MessageManagerService`. Returns an instance of the business objects message manager service.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Delete

```
Delete(System.Int64)
```

Deletes a private message from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the message to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxPrivateMessageIdLabel"
AssociatedControlID="uxPrivateMessageId" CssClass="span-3 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField ID="uxPrivateMessageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
using Ektron.Cms;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            long privateMessageId = long.Parse(uxPrivateMessageId.Text);

            PrivateMessageManager privateMessageManager = new PrivateMessageManager();
            PrivateMessageData messageData = privateMessageManager.GetItem
(privateMessageId);
            if (messageData != null)
            {
                privateMessageManager.Delete(privateMessageId);
                MessageUtilities.UpdateMessage(uxMessage, "Private message deleted",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter a Valid
PrivateMessageId", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific PrivateMessageData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the message to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Message ID="uxMessage" DisplayMode="Information" runat="server" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="Label2" AssociatedControlID="uxTo" CssClass="span-3 last"
runat="server" Text="To :" />
        <asp:Literal ID="uxTo" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="Label1" AssociatedControlID="uxFrom" CssClass="span-3 last"
runat="server" Text="From :" />
        <asp:Literal ID="uxFrom" runat="server"></asp:Literal>
    </li>

    <li class="clearfix">
        <ektronUI:Label ID="Label3" AssociatedControlID="uxSubject" CssClass="span-3
last" runat="server" Text="Subject :" />
        <asp:Literal ID="uxSubject" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="Label4" AssociatedControlID="uxFrom" CssClass="span-3 last"
runat="server" Text="Text :" />
        <asp:Literal ID="uxText" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long privateMessageId = long.Parse(uxPrivateMessageId.Text);

            PrivateMessageManager privateMessageManager = new PrivateMessageManager();
            PrivateMessageData messageData = privateMessageManager.GetItem
(privateMessageId);

            if (messageData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for private message
with ID " + messageData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxTo.Text = messageData.Recipients[0].ToUserDisplayName;
                uxFrom.Text = messageData.FromUserDisplayName;
                uxSubject.Text = messageData.Subject;
                uxText.Text = messageData.Message;
            }
```

```
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Private Message does not
exists. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Community.PrivateMessageCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Return Unread Only

Parameters

- `criteria`. Criteria used to retrieve messages.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
<li class="clearfix">
    <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxUnReadOnly"
CssClass="span-4 last" runat="server" Text="* Return Unread Only:" />
    <ektronUI:Button DisplayMode="Checkbox" Checked="false" Text="" ID="uxUnReadOnly"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
</li>
<li class="clearfix">
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</li>
</ol>

<asp:ListView ID="uxMessageListListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
```

```
<EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        From
                    </th>
                    <th>
                        Subject
                    </th>
                    <th>
                        Date
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("FromUserDisplayName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Subject")%>
            </td>
            <td class="devsite-method">
                <%# Eval("DateCreated")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PrivateMessageManager privateMessageManager = new PrivateMessageManager();
```

```
            if (privateMessageManager.UserId == 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Should be logged in to
retrieve private messages.", Message.DisplayModes.Error);
                uxPageMultiView.SetActiveView(uxViewMessage);
                return;
            }

            PrivateMessageCriteria criteria = new PrivateMessageCriteria();
            criteria.AddFilter(PrivateMessageProperty.RecipientUserId,
CriteriaFilterOperator.EqualTo, privateMessageManager.UserId);

            if (uxUnReadOnly.Checked)
            {
                criteria.AddFilter(PrivateMessageProperty.IsRead,
CriteriaFilterOperator.EqualTo, false);
            }

            List<PrivateMessageData> messageList = privateMessageManager.GetList
(criteria);

            uxMessageListListView.Visible = true;
            uxMessageListListView.DataSource = messageList;
            uxMessageListListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

MarkRead

```
MarkRead(System.Int64)
```

Marks the supplied message as read for the currently logged-in user.

NOTE: This method will not work in `ApiAccessMode.Admin` because a user is in charge of his or her own private messages.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `messageId`. ID of the message to mark as read.

.aspx code snippet

```
<div class="ektron-ui-form registerForm">
    <ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxPrivateMessageIdLabel"
AssociatedControlID="uxPrivateMessageId" CssClass="span-3 last" runat="server" Text="*
Id:" />
            <ektronUI:TextField ID="uxPrivateMessageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
        </li>
        <li class="clearfix">
            <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Mark As Read"></ektronUI:Button>
            <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server"
Text="* - Required" />
        </li>
    </ol>
</div>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long privateMessageId = long.Parse(uxPrivateMessageId.Text);

            PrivateMessageManager privateMessageManager = new PrivateMessageManager();
            PrivateMessageData messageData = privateMessageManager.GetItem
(privateMessageId);

            if (messageData != null)
            {

                privateMessageManager.MarkRead(privateMessageId);

                MessageUtilities.UpdateMessage(uxMessage, "Private Message Marked Read",
Message.DisplayModes.Success);
            }
```

```
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Private Message does not
exists. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

MarkUnread

```
MarkUnread(System.Int64)
```

Marks the supplied message as unread for the currently logged-in user.

NOTE: This method will not work in `ApiAccessMode.Admin` because a user is in charge of his or her own private messages.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `messageId`. ID of the message to mark as read.

.aspx code snippet

```
<div class="ektron-ui-form registerForm">
    <ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxPrivateMessageIdLabel"
AssociatedControlID="uxPrivateMessageId" CssClass="span-3 last" runat="server" Text="*
Id:" />
            <ektronUI:TextField ID="uxPrivateMessageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
        </li>
        <li class="clearfix">
            <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Mark As Unread"></ektronUI:Button>
            <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server"
Text="* - Required" />
        </li>
```

```
    </ol>
</div>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long privateMessageId = long.Parse(uxPrivateMessageId.Text);

            PrivateMessageManager privateMessageManager = new PrivateMessageManager();
            PrivateMessageData messageData = privateMessageManager.GetItem
(privateMessageId);

            if (messageData != null)
            {

                privateMessageManager.MarkUnread(privateMessageId);

                MessageUtilities.UpdateMessage(uxMessage, "Private Message Marked Read",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Private Message does not
exists. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Send

```
Send(Ektron.Cms.PrivateMessageData)
```

Sends a private message based on the supplied PrivateMessageData object. You can add multiple recipients using PrivateMessageData.Recipients property.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * To (User ID)
- * Subject
- * Message

Parameters

- `messageData`. The PrivateMessageData object to send.

Returns

Returns the PrivateMessageData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="Label1" AssociatedControlID="uxRecipientId" CssClass="span-4
last" runat="server" Text="* To (User Id):" />
        <ektronUI:TextField ID="uxRecipientId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSubjectLabel" AssociatedControlID="uxSubject"
CssClass="span-4 last" runat="server" Text="* Subject:" />
        <ektronUI:TextField ID="uxSubject" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTextLabel" AssociatedControlID="uxText" CssClass="span-4
last" runat="server" Text="* Message:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxText" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Send"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<ektronUI:Message ID="uxMessage" DisplayMode="Information" runat="server" />
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            long toUserId;

            if (!long.TryParse(uxRecipientId.Text, out toUserId)){
                MessageUtilities.UpdateMessage(uxMessage, "Valid To User ID required.",
Message.DisplayModes.Error);
                return;
            }

            PrivateMessageManager privateMessageManager = new PrivateMessageManager();
            PrivateMessageData privateMessageData = new PrivateMessageData()
            {
                Subject = uxSubject.Text,
                Message = uxText.Text
            };

            privateMessageData.AddRecipient(new PrivateMessageRecipientData() { ToUserID
= toUserId });

            privateMessageManager.Send(privateMessageData);

            MessageUtilities.UpdateMessage(uxMessage, "Private Message added with ID : "
+ privateMessageData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

PrivateMessageData

Namespace

```
Ektron.Cms
```

Properties

- AddRecipient(Ektron.Cms.PrivateMessageRecipientData)

  ```
  public void AddRecipient(Ektron.Cms.PrivateMessageRecipientData Recipient)
  ```

- DateCreated

  ```
  public System.DateTime DateCreated { set; get; }
  ```

- Ext_Values

  ```
  public int Ext_Values { set; get; }
  ```

- FolderId

  ```
  public long FolderID { set; get; }
  ```

- FromUserDeleted

  ```
  public bool FromUserDeleted { set; get; }
  ```

- FromUserDisplayName

  ```
  public string FromUserDisplayName { set; get; }
  ```

- FromUserID

  ```
  public long FromUserID { set; get; }
  ```

- ID

  ```
  public long ID { set; get; }
  ```

- Message

  ```
  public string Message { set; get; }
  ```

- Recipients

  ```
  public Ektron.Cms.PrivateMessageRecipientData[] Recipients { set; get; }
  ```

- SetID(long)

  ```
  public void SetID(long id)
  ```

- Subject

  ```
  public string Subject { set; get; }
  ```

RatingManager

8.60 and higher

The RatingManager class manages user ratings of content.

Namespace

```
Ektron.Cms.Framework.Community.RatingManager
```

Constructors

- `RatingManager()`
- `RatingManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RatingService`. Returns an instance of the business objects Rating Service.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.FriendTaxonomyData)
```

Adds a rating for the object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object ID
- * Object Type

- User Ratings (between 1 and 10 and no decimal values)
- User Comments

Parameters

- `ratingData`. The RatingData object to add.

Returns

Returns the custom RatingData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectIDLabel" AssociatedControlID="uxObjectID"
CssClass="span-4 last"runat="server" Text="* Object Id:" />
        <ektronUI:TextField ID="uxObjectID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"Text="0" />
    </li>

    <li class="clearfix">
        <ektronUI:Label ID="uxObjectTypeLabel" AssociatedControlID="uxObjectType"
CssClass="span-5 last"runat="server" Text="* Object Type:" />
        <asp:DropDownList ID="uxObjectType" runat="server">
            <asp:ListItem>Content</asp:ListItem>
            <asp:ListItem>Folder</asp:ListItem>
            <asp:ListItem>User</asp:ListItem>
            <asp:ListItem>UserGroup</asp:ListItem>
            <asp:ListItem>Library</asp:ListItem>
            <asp:ListItem>Collection</asp:ListItem>
            <asp:ListItem>Calendar</asp:ListItem>
            <asp:ListItem>Subscription</asp:ListItem>
            <asp:ListItem>Notification</asp:ListItem>
            <asp:ListItem>Blog</asp:ListItem>
            <asp:ListItem>BlogPost</asp:ListItem>
            <asp:ListItem>BlogComment</asp:ListItem>
            <asp:ListItem>DiscussionBoard</asp:ListItem>
            <asp:ListItem>DiscussionForum</asp:ListItem>
            <asp:ListItem>RestrictIP</asp:ListItem>
            <asp:ListItem>ReplaceWord</asp:ListItem>
            <asp:ListItem>UserRank</asp:ListItem>
            <asp:ListItem>DiscussionTopic</asp:ListItem>
            <asp:ListItem>CommunityGroup</asp:ListItem>
            <asp:ListItem>TaxonomyNode</asp:ListItem>
            <asp:ListItem>CatalogEntry</asp:ListItem>
            <asp:ListItem>MessageBoard</asp:ListItem>
            <asp:ListItem>CalendarEvent</asp:ListItem>
            <asp:ListItem>MicroMessage</asp:ListItem>
            <asp:ListItem>Subscriber</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserRatingLabel" AssociatedControlID="uxUserRating"
CssClass="span-4 last"runat="server" Text="User Rating(between 1 and 10 and no decimal
values):" />
```

```
        <ektronUI:TextField ID="uxUserRating" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserCommentsLabel" AssociatedControlID="uxUserComments"
CssClass="span-4 last"runat="server" Text="User Comments:" />
        <ektronUI:TextField ID="uxUserComments" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Common;
using Ektron.Cms;
using Microsoft.VisualBasic;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RatingManager ratingManager = new RatingManager();
            RatingData ratingData = new RatingData();

            long objectID = long.Parse(uxObjectID.Text);
            bool idResult = Information.IsNumeric(uxObjectID.Text);

            int Rating = int.Parse(uxUserRating.Text);
            bool cRatingResult = Information.IsNumeric(uxObjectID.Text);

            if (!(cRatingResult))
            {
                uxMessage.Text = " Rating Should be between 0 and 10";
                return;
            }

            string userComments = uxUserComments.Text;
            UserAPI userApi = new UserAPI();
            long UserId = userApi.RequestInformationRef.UserId;


            EkEnumeration.CMSObjectTypes objectType =
(EkEnumeration.CMSObjectTypes)Enum.Parse(typeof(EkEnumeration.CMSObjectTypes),
```

```
uxObjectType.Text, true);

            ratingData.ObjectId = objectID;
            ratingData.ObjectType = objectType;
            ratingData.LanguageId = ratingData.LanguageId;
            ratingData.UserComments = userComments;
            ratingData.UserId = UserId;
            ratingData.UserRating = Rating;
            ratingData.VisitorId = userApi.RequestInformationRef.ClientEktGUID;
            ratingManager.Add(ratingData);
            uxPageMultiView.SetActiveView(uxViewMessage);

            MessageUtilities.UpdateMessage(uxMessage, "Ratting Added with RatingId: " +
ratingData.Id, Message.DisplayModes.Success);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Delete

```
Delete(System.Int64)
```

Deletes a rating.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Rating ID

Parameters

- `ratingId`. The identifier of the rating to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRatingIdLabel" AssociatedControlID="uxRatingId"
CssClass="span-3 last" runat="server" Text="* Rating Id:" />
        <ektronUI:TextField ID="uxRatingId" CssClass="span-6" runat="server"
```

```
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRatingStatus" Visible="false" CssClass="span-3 last"
runat="server" Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
using Microsoft.VisualBasic;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
   {
       try
       {
           RatingManager ratingManager = new RatingManager();
           RatingData ratingData = new RatingData();
           long ratingID = long.Parse(uxRatingId.Text);
           bool result = Information.IsNumeric(uxRatingId.Text);
           ratingData = ratingManager.GetItem(ratingID);
           if (!((result) && ratingData != null))
           {
               uxStatus.Visible = true;
               MessageUtilities.UpdateMessage(uxStatus, "Invalid Rating ID ",
Message.DisplayModes.Error);
               return;
           }

           ratingManager.Delete(ratingID);

           MessageUtilities.UpdateMessage(uxMessage, "The following rating ID
successfully deleted  : " + ratingID, Message.DisplayModes.Success);

           uxPageMultiView.SetActiveView(uxViewMessage);
       }
       catch (Exception ex)
       {
           MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the rating object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Rating ID

Parameters

- `ratingId`. The ID of the rating item to get.

Returns

Returns the rating object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRatingIdLabel" AssociatedControlID="uxRatingId"
CssClass="span-3 last" runat="server" Text="* Rating Id:" />
        <ektronUI:TextField ID="uxRatingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetDetails"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
```

```
    <li class="clearfix">
        <asp:Literal ID="uxID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLanguage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserRating" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxRatingDate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserComment" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Community;
using Ektron.Cms.Framework.Community;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            RatingManager ratingManager = new RatingManager();
            RatingData ratingData = new RatingData();
            long ratingID = long.Parse(uxRatingId.Text);
            bool result = Information.IsNumeric(uxRatingId.Text);
            ratingData = ratingManager.GetItem(ratingID);
            if (!((result) && ratingData != null))
            {
                MessageUtilities.UpdateMessage(uxMessage, "Invalid  Rating ID ",
Message.DisplayModes.Error);
                uxPageMultiView.SetActiveView(uxViewMessage);
                return;
            }

            MessageUtilities.UpdateMessage(uxMessage, " Rating details for   rating Id :
" + ratingID + " are below", Message.DisplayModes.Success);
            uxID.Text = " ID : " + ratingData.Id;
            uxLanguage.Text = "ContentLanguage Id : " + ratingData.LanguageId;
            uxUserID.Text = " Reviewed UserId : " + ratingData.UserId;
```

```
            uxUserRating.Text = "User Rating : " + ratingData.UserRating;
            uxRatingDate.Text = "RatingDate : " + ratingData.RatingDate;
            uxUserComment.Text = "UserComments : " + ratingData.UserComments;
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }


    }
```

GetList

```
GetList(Ektron.Cms.Community.RatingCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Rating Property
- * Object Value

Parameters

- `criteria`. RatingCriteria used to retrieve objects.

Returns

A list of RatingData.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRatingPropertyLabel"
AssociatedControlID="uxRatingProperty"
                    CssClass="span-6 last" runat="server" Text="* RatingProperty:" />
        <asp:DropDownList ID="uxRatingProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>LanguageId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last"
                    runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxRatingListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
            Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.
    </EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Rating
                    </th>
                    <th>
                        User Comments
                    </th>
                    <th>
                        Rating Date
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserRating")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserComments")%>
            </td>
```

```
            <td class="devsite-method">
                <%# Eval("RatingDate")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            RatingCriteria ratingCriteria = new RatingCriteria(RatingProperty.Id,
EkEnumeration.OrderByDirection.Descending);
            if (uxRatingProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                ratingCriteria.AddFilter(RatingProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = int.Parse(uxObjectValue.Text);
                ratingCriteria.AddFilter(RatingProperty.ObjectId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            RatingManager ratingManager = new RatingManager();
            List<RatingData> ratingList = ratingManager.GetList(ratingCriteria);
            if (ratingList.Count > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, " Rating details are below",
Message.DisplayModes.Success);
                uxRatingListView.Visible = true;
                uxRatingListView.DataSource = ratingList;
                uxRatingListView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Rating details not available
```

```
for the Criteria", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetUserTaxonomy

```
GetStatistics(System.Int64,Ektron.Cms.Common.EkEnumeration.CMSObjectTypes)
```

Retrieves a list of statistics by object type and ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object ID
- * Object Type

Parameters

- `objectType`. Type of the object has rated.
- `objectId`. Rated object ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectIdLabel"
AssociatedControlID="uxObjectId"CssClass="span-4 last" runat="server" Text="* Object
Id:"/>
        <ektronUI:TextField ID="uxObjectId" CssClass="span-6"runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxObjectTypeLabel"
AssociatedControlID="uxObjectType"CssClass="span-5 last" runat="server" Text="* Object
Type:"/>
        <asp:DropDownList ID="uxObjectType" runat="server">
            <asp:ListItem>Content</asp:ListItem>
            <asp:ListItem>Folder</asp:ListItem>
            <asp:ListItem>User</asp:ListItem>
            <asp:ListItem>UserGroup</asp:ListItem>
            <asp:ListItem>Library</asp:ListItem>
```

```
            <asp:ListItem>Collection</asp:ListItem>
            <asp:ListItem>Calendar</asp:ListItem>
            <asp:ListItem>Subscription</asp:ListItem>
            <asp:ListItem>Notification</asp:ListItem>
            <asp:ListItem>Blog</asp:ListItem>
            <asp:ListItem>BlogPost</asp:ListItem>
            <asp:ListItem>BlogComment</asp:ListItem>
            <asp:ListItem>DiscussionBoard</asp:ListItem>
            <asp:ListItem>DiscussionForum</asp:ListItem>
            <asp:ListItem>RestrictIP</asp:ListItem>
            <asp:ListItem>ReplaceWord</asp:ListItem>
            <asp:ListItem>UserRank</asp:ListItem>
            <asp:ListItem>DiscussionTopic</asp:ListItem>
            <asp:ListItem>CommunityGroup</asp:ListItem>
            <asp:ListItem>TaxonomyNode</asp:ListItem>
            <asp:ListItem>CatalogEntry</asp:ListItem>
            <asp:ListItem>MessageBoard</asp:ListItem>
            <asp:ListItem>CalendarEvent</asp:ListItem>
            <asp:ListItem>MicroMessage</asp:ListItem>
            <asp:ListItem>Subscriber</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit"runat="server" OnClick="uxSubmit_Click" Text="Get
Statistics"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server"Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage"runat="server">
</asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Common;
using Microsoft.VisualBasic;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            long ObjectID = long.Parse(uxObjectId.Text);
            bool result = Information.IsNumeric(ObjectID);
            RatingManager ratingManager = new RatingManager();
            EkEnumeration.CMSObjectTypes objectType =
(EkEnumeration.CMSObjectTypes)Enum.Parse(typeof(EkEnumeration.CMSObjectTypes),
uxObjectType.Text, true);
            RatingStatisticData ratingstatisticsdata = ratingManager.GetStatistics
```

```
(ObjectID, objectType);

            MessageUtilities.UpdateMessage(uxMessage, "The rating statistics data for
the objectId : " +  ObjectID , Message.DisplayModes.Success);
            uxRatingID.Text = " ID : " + ratingstatisticsdata.Id;
            uxRatingObjectId.Text = " Rating Object Id : " +
ratingstatisticsdata.ObjectId;
            uxRatingObjectType.Text = " Rating Object TYpe : " +
ratingstatisticsdata.ObjectType;
            uxUserRatingAverage.Text = " Rating Average : " +
ratingstatisticsdata.RatingAverage;
            uxRatingCount.Text = " Rating Count : " + ratingstatisticsdata.RatingCount;
            uxUserRatingSum.Text = " Rating Sum : " + ratingstatisticsdata.RatingSum;
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Purge

```
Purge
(System.Int64,Ektron.Cms.Common.EkEnumeration.CMSObjectTypes,System.DateTime,System.Date
Time)
```

Purges all ratings of a particular object ID and type.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object ID
- * Object Type
- * Start Date and Time
- * End Date and Time

Parameters

- `objectId`. The ID of the object that is having its ratings purged.
- `objectType`. The Type of the object that is having its ratings purged.
- `startDate`. The start date of object that is having its ratings purged.
- `endDate`. The end date of object that is having its ratings purged.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectIdLabel"
AssociatedControlID="uxObjectId"CssClass="span-4 last" runat="server" Text="* Object
Id:"/>
        <ektronUI:TextField ID="uxObjectId" CssClass="span-6"runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectTypeLabel"
AssociatedControlID="uxObjectType"CssClass="span-5 last" runat="server" Text="* Object
Type:"/>
        <asp:DropDownList ID="uxObjectType" runat="server">
            <asp:ListItem>Content</asp:ListItem>
            <asp:ListItem>Folder</asp:ListItem>
            <asp:ListItem>User</asp:ListItem>
            <asp:ListItem>UserGroup</asp:ListItem>
            <asp:ListItem>Library</asp:ListItem>
            <asp:ListItem>Collection</asp:ListItem>
            <asp:ListItem>Calendar</asp:ListItem>
            <asp:ListItem>Subscription</asp:ListItem>
            <asp:ListItem>Notification</asp:ListItem>
            <asp:ListItem>Blog</asp:ListItem>
            <asp:ListItem>BlogPost</asp:ListItem>
            <asp:ListItem>BlogComment</asp:ListItem>
            <asp:ListItem>DiscussionBoard</asp:ListItem>
            <asp:ListItem>DiscussionForum</asp:ListItem>
            <asp:ListItem>RestrictIP</asp:ListItem>
            <asp:ListItem>ReplaceWord</asp:ListItem>
            <asp:ListItem>UserRank</asp:ListItem>
            <asp:ListItem>DiscussionTopic</asp:ListItem>
            <asp:ListItem>CommunityGroup</asp:ListItem>
            <asp:ListItem>TaxonomyNode</asp:ListItem>
            <asp:ListItem>CatalogEntry</asp:ListItem>
            <asp:ListItem>MessageBoard</asp:ListItem>
            <asp:ListItem>CalendarEvent</asp:ListItem>
            <asp:ListItem>MicroMessage</asp:ListItem>
            <asp:ListItem>Subscriber</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
          <ektronUI:Label ID="Label4" AssociatedControlID="uxStartDate" CssClass="span-3
last"
              runat="server" Text="* Start DateTime:" />
          <ektronUI:DatePicker Rows="3" ID="DatePicker1" CssClass="span-6"
runat="server" Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
          <asp:DropDownList ID="DropDownList2" runat="server">
              <asp:ListItem>12</asp:ListItem>
              <asp:ListItem>1</asp:ListItem>
              <asp:ListItem>2</asp:ListItem>
              <asp:ListItem>3</asp:ListItem>
              <asp:ListItem>4</asp:ListItem>
              <asp:ListItem>5</asp:ListItem>
              <asp:ListItem>6</asp:ListItem>
```

```
                <asp:ListItem>7</asp:ListItem>
                <asp:ListItem>8</asp:ListItem>
                <asp:ListItem>9</asp:ListItem>
                <asp:ListItem>10</asp:ListItem>
                <asp:ListItem>11</asp:ListItem>
            </asp:DropDownList>
            <asp:DropDownList ID="DropDownList3" runat="server">
                <asp:ListItem>AM</asp:ListItem>
                <asp:ListItem>PM</asp:ListItem>
            </asp:DropDownList>
        </li>
        <li class="clearfix">
            <ektronUI:Label ID="Label5" AssociatedControlID="uxEndDate" CssClass="span-3
last"
                runat="server" Text="* End DateTime:" />
            <ektronUI:DatePicker Rows="3" ID="DatePicker2" CssClass="span-6"
runat="server" Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
            <asp:DropDownList ID="DropDownList4" runat="server">
                <asp:ListItem>1</asp:ListItem>
                <asp:ListItem>2</asp:ListItem>
                <asp:ListItem>3</asp:ListItem>
                <asp:ListItem>4</asp:ListItem>
                <asp:ListItem>5</asp:ListItem>
                <asp:ListItem>6</asp:ListItem>
                <asp:ListItem>7</asp:ListItem>
                <asp:ListItem>8</asp:ListItem>
                <asp:ListItem>9</asp:ListItem>
                <asp:ListItem>10</asp:ListItem>
                <asp:ListItem>11</asp:ListItem>
                <asp:ListItem>12</asp:ListItem>
            </asp:DropDownList>
            <asp:DropDownList ID="DropDownList5" runat="server">
                <asp:ListItem>AM</asp:ListItem>
                <asp:ListItem>PM</asp:ListItem>
            </asp:DropDownList>
        </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit"runat="server" OnClick="uxSubmit_Click"
Text="Purge"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server"Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage"runat="server">
</asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
using Ektron.Cms.Common;
```

```
using Microsoft.VisualBasic;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;
            String et = uxEndDate.Text + " " + uxEndTime.SelectedItem.Text + ":00:00 " +
uxEndPeriod.SelectedItem.Text;
            DateTime startDateTime = DateTime.Parse(st);
            DateTime endDateTime = DateTime.Parse(et);

            long ObjectID = long.Parse(uxObjectId.Text);
            RatingManager ratingManager = new RatingManager();
            EkEnumeration.CMSObjectTypes objectType =
(EkEnumeration.CMSObjectTypes)Enum.Parse(typeof(EkEnumeration.CMSObjectTypes),
uxObjectType.Text, true);

            ratingManager.Purge(ObjectID, objectType, startDateTime, endDateTime);

            MessageUtilities.UpdateMessage(uxMessage, "The rating successfully purged.",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Purge (by state)

```
Purge
(System.Int64,Ektron.Cms.Common.EkEnumeration.CMSObjectTypes,System.DateTime,System.Date
Time,Ektron.Cms.Common.EkEnumeration.RatingState)
```

Purges all ratings of a particular object ID and type.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object ID
- * Object Type

- * Start Date and Time
- * End Date and Time
- * Rating State (Pending, Approved, Rejected)

Parameters

- `objectId`. The ID of the object that is having its ratings purged.
- `objectType`. The Type of the object that is having its ratings purged.
- `startDate`. The start date of object that is having its ratings purged.
- `endDate`. The end date of object that is having its ratings purged.
- `ratingState`. The rating state of object that is having its ratings purged.

.aspx code snippet

```
<ol class="formFields">
      <li class="clearfix">
          <ektronUI:Label ID="Label1" AssociatedControlID="uxObjectId" CssClass="span-3
last"
              runat="server" Text="* Object Id:" />
          <ektronUI:TextField ID="TextField1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
      </li>
      <li class="clearfix">
          <ektronUI:Label ID="Label2" Visible="false" CssClass="span-3 last"
runat="server"
              Text="Status:" />
      </li>
      <li class="clearfix">
          <ektronUI:Label ID="Label3" AssociatedControlID="uxObjectType" CssClass="span-
3 last"
              runat="server" Text="* Object Type:" />
          <asp:DropDownList ID="DropDownList1" runat="server">
              <asp:ListItem>Content</asp:ListItem>
              <asp:ListItem>Folder</asp:ListItem>
              <asp:ListItem>User</asp:ListItem>
              <asp:ListItem>UserGroup</asp:ListItem>
              <asp:ListItem>Library</asp:ListItem>
              <asp:ListItem>Collection</asp:ListItem>
              <asp:ListItem>Calendar</asp:ListItem>
              <asp:ListItem>Subscription</asp:ListItem>
              <asp:ListItem>Notification</asp:ListItem>
              <asp:ListItem>Blog</asp:ListItem>
              <asp:ListItem>BlogPost</asp:ListItem>
              <asp:ListItem>BlogComment</asp:ListItem>
              <asp:ListItem>DiscussionBoard</asp:ListItem>
              <asp:ListItem>DiscussionForum</asp:ListItem>
              <asp:ListItem>RestrictIP</asp:ListItem>
              <asp:ListItem>ReplaceWord</asp:ListItem>
              <asp:ListItem>UserRank</asp:ListItem>
              <asp:ListItem>DiscussionTopic</asp:ListItem>
              <asp:ListItem>CommunityGroup</asp:ListItem>
              <asp:ListItem>TaxonomyNode</asp:ListItem>
              <asp:ListItem>CatalogEntry</asp:ListItem>
```

```
            <asp:ListItem>MessageBoard</asp:ListItem>
            <asp:ListItem>CalendarEvent</asp:ListItem>
            <asp:ListItem>MicroMessage</asp:ListItem>
            <asp:ListItem>Subscriber</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="Label4" AssociatedControlID="uxStartDate" CssClass="span-3
last"
            runat="server" Text="* Start DateTime:" />
        <ektronUI:DatePicker Rows="3" ID="DatePicker1" CssClass="span-6"
runat="server" Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="DropDownList2" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="DropDownList3" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="Label5" AssociatedControlID="uxEndDate" CssClass="span-3
last"
            runat="server" Text="* End DateTime:" />
        <ektronUI:DatePicker Rows="3" ID="DatePicker2" CssClass="span-6"
runat="server" Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="DropDownList4" runat="server">
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
            <asp:ListItem>12</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="DropDownList5" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
```

```
      <li class="clearfix">
            <ektronUI:Label ID="Label1" AssociatedControlID="uxEndDate" CssClass="span-3
last"
                runat="server" Text="* Rating State:" />
            <asp:DropDownList ID="DropDownList1" runat="server">
                <asp:ListItem>Pending</asp:ListItem>
                <asp:ListItem>Approved</asp:ListItem>
                <asp:ListItem>Rejected</asp:ListItem>
            </asp:DropDownList>
      </li>
      <li class="clearfix">
          <ektronUI:Button ID="Button1" runat="server" OnClick="uxSubmit_Click"
Text="Purge"></ektronUI:Button>
          <ektronUI:Label ID="Label6" CssClass="span-4" runat="server" Text="* -
Required" />
      </li>
  </ol>
  <asp:Literal ID="uxMessage"runat="server">
</asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
using Microsoft.VisualBasic;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            long ObjectID = long.Parse(uxObjectId.Text);
            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;
            String et = uxEndDate.Text + " " + uxEndTime.SelectedItem.Text + ":00:00 " +
uxEndPeriod.SelectedItem.Text;
            DateTime startDateTime = DateTime.Parse(st);
            DateTime endDateTime = DateTime.Parse(et);

            RatingManager ratingManager = new RatingManager();
            EkEnumeration.CMSObjectTypes objectType =
(EkEnumeration.CMSObjectTypes)Enum.Parse(typeof(EkEnumeration.CMSObjectTypes),
uxObjectType.Text, true);
            EkEnumeration.RatingState ratingState =
(EkEnumeration.RatingState)Enum.Parse(typeof(EkEnumeration.RatingState),
uxRatingState.SelectedItem.Text, true);

            ratingManager.Purge(ObjectID, objectType, startDateTime, endDateTime,
ratingState);
            MessageUtilities.UpdateMessage(uxMessage, "The rating successfully purged.",
```

```
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.RatingData)
```

Updates a rating for an object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Rating ID
- * Object Type
- User Rating (between 1 and 10 and no decimal values)
- User Comments

Parameters

- `ratingData`. The rating object to update.

Returns

Returns the custom RatingData object updated.

Remarks

Validate the rating item with `GetItem()` before you update the properties. Then, call `Update` with your modified RatingData object.

NOTE: You cannot change all properties after the initial Add event, such as the RatingData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRatingIdLabel" AssociatedControlID="uxRatingId"
                    CssClass="span-3 last" runat="server" Text="* Rating Id:" />
        <ektronUI:TextField ID="uxRatingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
```

```
                    Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectTypeLabel" AssociatedControlID="uxObjectType"
CssClass="span-5 last"
                    runat="server" Text="* Object Type:" />
        <asp:DropDownList ID="uxObjectType" runat="server">
            <asp:ListItem>Content</asp:ListItem>
            <asp:ListItem>Folder</asp:ListItem>
            <asp:ListItem>User</asp:ListItem>
            <asp:ListItem>UserGroup</asp:ListItem>
            <asp:ListItem>Library</asp:ListItem>
            <asp:ListItem>Collection</asp:ListItem>
            <asp:ListItem>Calendar</asp:ListItem>
            <asp:ListItem>Subscription</asp:ListItem>
            <asp:ListItem>Notification</asp:ListItem>
            <asp:ListItem>Blog</asp:ListItem>
            <asp:ListItem>BlogPost</asp:ListItem>
            <asp:ListItem>BlogComment</asp:ListItem>
            <asp:ListItem>DiscussionBoard</asp:ListItem>
            <asp:ListItem>DiscussionForum</asp:ListItem>
            <asp:ListItem>RestrictIP</asp:ListItem>
            <asp:ListItem>ReplaceWord</asp:ListItem>
            <asp:ListItem>UserRank</asp:ListItem>
            <asp:ListItem>DiscussionTopic</asp:ListItem>
            <asp:ListItem>CommunityGroup</asp:ListItem>
            <asp:ListItem>TaxonomyNode</asp:ListItem>
            <asp:ListItem>CatalogEntry</asp:ListItem>
            <asp:ListItem>MessageBoard</asp:ListItem>
            <asp:ListItem>CalendarEvent</asp:ListItem>
            <asp:ListItem>MicroMessage</asp:ListItem>
            <asp:ListItem>Subscriber</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <asp:Label ID="Label1" Visible="false" CssClass="span-3 last" runat="server"
Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserRatingLabel" AssociatedControlID="uxUserRating"
CssClass="span-3 last"
                    runat="server" Text="* User Rating :  <br />(between 1 and 10 and no
decimal values)" />
        <ektronUI:TextField ID="uxUserRating" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
                    Text="9" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserCommentsLabel" AssociatedControlID="uxUserComments"
CssClass="span-3 last"
                    runat="server" Text="User Comments:" />
        <ektronUI:TextField ID="uxUserComments" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
                    Text="" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button ID="Button1" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="Label2" CssClass="span-4" runat="server" Text="* - Required"
/>
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System;
using Ektron.Cms.Framework.Community;
using Ektron.Cms;
using Ektron.Cms.Common;
using Microsoft.VisualBasic;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RatingManager ratingManager = new RatingManager();
            RatingData ratingData = new RatingData();
            long ratingID = long.Parse(uxRatingId.Text);
            bool result = Information.IsNumeric(uxRatingId.Text);
            ratingData = ratingManager.GetItem(ratingID);

            if (!((result) && ratingData != null))
            {
                uxMessage.Text = "Invalid Rating ID ";
                uxPageMultiView.SetActiveView(uxViewMessage);
                return;
            }

            int Rating = int.Parse(uxUserRating.Text);
            bool ratingResult = Information.IsNumeric(Rating);

            if (!(ratingResult))
            {
                MessageUtilities.UpdateMessage(uxMessage, "Rating Should be between 0
and 10", Message.DisplayModes.Error);
                return;
            }

            string userComments = uxUserComments.Text;
            UserAPI userApi = new UserAPI();
            long UserId = userApi.RequestInformationRef.UserId;
            EkEnumeration.CMSObjectTypes objectType =
(EkEnumeration.CMSObjectTypes)Enum.Parse(typeof(EkEnumeration.CMSObjectTypes),
uxObjectType.Text, true);

            ratingData.UserComments = userComments;
```

```
            ratingData.ObjectType = objectType;
            ratingData.UserId = UserId;
            ratingData.UserRating = Rating;
            ratingData.VisitorId = userApi.RequestInformationRef.ClientEktGUID;
            ratingManager.Update(ratingData);
            MessageUtilities.UpdateMessage(uxMessage, "Rating Updated with Rating ID : "
+ ratingData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Data Classes

RatingCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Community
```

Constructors

- `RatingCriteria()`

  ```
  public RatingCriteria()
  ```

- `RatingCriteria(Ektron.Cms.Common.RatingProperty, EkEnumeration.OrderByDirection)`

  ```
  public RatingCriteria(Ektron.Cms.Common.RatingProperty orderByField,
  EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `RatingProperty` are:

  - Id
  - LanguageId
  - ObjectId
  - ObjectType
  - RatingDate
  - RatingState
  - UserId
  - VisitorId

RatingData

Namespace

```
Ektron.Cms
```

Properties

- LanguageId

```
public int LanguageId { set; get; }
```

- ObjectId

```
public long ObjectId { set; get; }
```

- ObjectType

```
public Ektron.Cms.Common.EkEnumeration.CMSObjectTypes
   ObjectType { set; get; }
```

- RatingDate

```
public System.DateTime RatingDate { set; get; }
```

- RatingState

```
public Ektron.Cms.Common.EkEnumeration.RatingState
  RatingState { set; get; }
```

- UserComments

```
public string UserComments { set; get; }
```

- UserId

```
public long UserId { set; get; }
```

- UserRating

```
public int UserRating { set; get; }
```

- VisitorId

```
public string VisitorId { set; get; }
```

TagManager

8.50 and higher

The TagManager class manages tags. Tags are keywords that you can assign to content and library items, which allows for tag-based searching. For example, you can add the tag *Electronics* and tag content that is related to the electronic devices, so that people can search for the content using the *Electronics* tag. For information about how tagging is used in Ektron, see Tagging Content, Library Items, Users, and Groups with Keywords in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Community.TagManager
```

Constructors

- `TagManager()`
- `TagManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `TagManagerService`. Returns an instance of the business objects TagManager service.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* on the facing page
- *Delete* on page 1044
- *GetItem* on page 1046
- *GetList* on page 1048

- *Tag* on page 1054
- *Update* on page 1055

Add

```
Add(Ektron.Cms.TagData)
```

Adds a tag based on information in a TagData object. The method populates the tag.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Tag Type
- * Tag Text

Parameters

- `tagData`. The TagData object to add.

Returns

Returns the custom TagData object added.

.aspx Code Snippxet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTagTypeLabel" AssociatedControlID="uxTagType"
CssClass="span-4 last" runat="server" Text="* Tag Type:" />
        <asp:DropDownList ID="uxTagType" runat="server">
            <asp:ListItem>ContentDefault</asp:ListItem>
            <asp:ListItem>UserDefault</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTagTextLabel" AssociatedControlID="uxTagText"
CssClass="span-4 last" runat="server" Text="* Tag Text:" />
        <ektronUI:TextField ID="uxTagText" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"/>
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            TagManager tagManager = new TagManager();
            TagData tagData = new TagData()
            {
                Type = uxTagType.Text == "ContentDefault" ? TagTypes.ContentDefault :
TagTypes.UserDefault ,
                Text = uxTagText.Text
            };

            tagManager.Add(tagData);

            MessageUtilities.UpdateMessage(uxMessage, "Tag added with Id " + tagData.Id,
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a tag from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the tag to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTagIdLabel" AssociatedControlID="uxTagId" CssClass="span-3
last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTagId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Community;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long tagId = long.Parse(uxTagId.Text);
            TagManager tagManager = new TagManager();
            TagData tagData = tagManager.GetItem(tagId);
            if (tagData != null)
            {
                tagManager.Delete(tagId);
                MessageUtilities.UpdateMessage(uxMessage, "Tag deleted.",
```

```
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Tag Id doesn't exists.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific TagData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. Tag's id to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTagIdLabel" AssociatedControlID="uxTagId" CssClass="span-3
last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxTagId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
```

```
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLanguageId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long tagId = long.Parse(uxTagId.Text);

            TagManager tagManager = new TagManager();
            TagData tagData = tagManager.GetItem(tagId);

            if (tagData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for tag with ID " +
tagData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxName.Text = "Text : " + tagData.Text ;
                uxType.Text = "Type : " + tagData.Type;
                uxLanguageId.Text = "Language Id : " + tagData.LanguageId;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Tag does not exist. ",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

GetList

```
GetList(Content.TagCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Tag Data Property
- * Object Value

Parameters

- `criteria`. TagCriteria used to retrieve tags.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTagDataPropertyLabel"
AssociatedControlID="uxTagDataProperty" CssClass="span-4 last" runat="server" Text="*
TagDataProperty:" />
        <asp:DropDownList ID="uxTagDataProperty" runat="server">
            <asp:ListItem>Type(0,1,2,4,8)</asp:ListItem>
            <asp:ListItem>Text</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxTagDataListListView" runat="server"
```

```
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Type
                    </th>
                    <th>
                        Text
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Type")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Text")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            object Objectvalue;

            Ektron.Cms.Community.TagCriteria criteria = new
Ektron.Cms.Community.TagCriteria();
            if (uxTagDataProperty.SelectedItem.Text == "Type(0,1,2,4,8)")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(TagProperty.Type, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TagProperty.Text, CriteriaFilterOperator.Contains,
Objectvalue);
            }

            TagManager tagManager = new TagManager();
            List<TagData> tagDataList = tagManager.GetList(criteria);

            uxTagDataListListView.Visible = true;
            uxTagDataListListView.DataSource = tagDataList;
            uxTagDataListListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTagCloud

```
GetTagCloud(TagCloudRequestData.Int32)
```

Retrieves a tag cloud.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Language ID (default: 1033)
- * Page Size (default: 50)
- * Tag Type (default: TagTypes.All)
- * Order By (default: TagOrderBy.Text)

- * Order By Direction (default: EkEnumeration.OrderByDirection.Ascending)
- * Object Type (default: EkEnumeration.CMSObjectTypes.User)

Parameters

- `request`. Contains the parameters for the tag cloud to retrieve.
- `totalRecords`. Returns the total number of tags available in the requested cloud.

Returns

A tag cloud object representing the request.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectTagDataPropertyLabel"
AssociatedControlID="uxObjectTagDataProperty"
            CssClass="span-3 last" runat="server" Text="* Tag ObjectType:" />
        <asp:DropDownList ID="uxObjectTagDataProperty" runat="server">
            <asp:ListItem>Content</asp:ListItem>
            <asp:ListItem>Library</asp:ListItem>
            <asp:ListItem>User</asp:ListItem>
        </asp:DropDownList>
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageIdLabel" AssociatedControlID="uxTagLanguageId"
CssClass="span-3 last"
            runat="server" Text="*Tag LanguageId :" />
        <ektronUI:TextField ID="uxTagLanguageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1033" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxTotalPagesLabel" AssociatedControlID="uxTotalPages"
CssClass="span-3 last"
            runat="server" Text="* TotalPages :" />
        <ektronUI:TextField ID="uxTotalPages" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="10" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
<li class="clearfix">
```

```
    <ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
</li>
<asp:ListView ID="uxObjectTagListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
<EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
<LayoutTemplate>
    <table class="devsite-api-method">
        <thead>
            <tr>
                <th>
                    Id
                </th>
                <th>
                    Language Id
                </th>
                <th>
                    Text
                </th>
                    <th>
                    TotalUsedCount
                </th>
            </tr>
        </thead>
        <tbody>
            <asp:PlaceHolder ID="aspItemPlaceholder" runat="server"></asp:PlaceHolder>
        </tbody>
    </table>
</LayoutTemplate>
<ItemTemplate>
    <tr>
        <td class="devsite-method">
            <%# Eval("Id")%>
        </td>
        <td class="devsite-method">
            <%# Eval("LanguageId")%>
        </td>
        <td class="devsite-method">
            <%# Eval("Text")%>
        </td>
        <td class="devsite-method">
            <%# Eval("TotalUsedCount")%>
        </td>
    </tr>
</ItemTemplate>
</asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Community;
using System.Collections.ObjectModel;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TagManager tagManager = new TagManager();
            int totalPages = int.Parse(uxTotalPages.Text);

            TagCloudRequestData tagCloudRequestData = new TagCloudRequestData();
            tagCloudRequestData.TagTypes = TagTypes.All;
            tagCloudRequestData.PageSize = 50;

            if (uxObjectTagDataProperty.SelectedItem.Text == "Content")
                tagCloudRequestData.ObjectType =
Ektron.Cms.Common.EkEnumeration.CMSObjectTypes.Content;
            else if (uxObjectTagDataProperty.SelectedItem.Text == "Library")
                tagCloudRequestData.ObjectType =
Ektron.Cms.Common.EkEnumeration.CMSObjectTypes.Library;
            else
                tagCloudRequestData.ObjectType =
Ektron.Cms.Common.EkEnumeration.CMSObjectTypes.User;

            tagCloudRequestData.LanguageId = int.Parse(uxTagLanguageId.Text);

            TagCloud tagCloud = tagManager.GetTagCloud(tagCloudRequestData, out
totalPages);
            Collection<TagCloudItem> tags = tagCloud.Tags;
            List<TagData> tagList = new List<TagData>();
            foreach (TagCloudItem tagItem in tags)
            {
                TagData tag = tagItem.Tag;
                if (tag != null)
                {
                    tagList.Add(tag);
                }
            }

            uxObjectTagListView.Visible = true;
            uxObjectTagListView.DataSource = tagList;
            uxObjectTagListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Tag

```
Tag(System.String,System.Int64,Ektron.Cms.Common.EkEnumeration.CMSObjectTypes)
```

Tags content with supplied details.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Tag
- * Object ID
- * Tag Type

Parameters

- `tag`. The string that contains the tag text.
- `objectId`. The object ID to be tagged.
- `objectType`. The object type to be tagged.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTagTextLabel" AssociatedControlID="uxTagText"
CssClass="span-4 last" runat="server" Text="* Tag:" />
        <ektronUI:TextField ID="uxTagText" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"/>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectIdLabel" AssociatedControlID="uxObjectId"
CssClass="span-4 last" runat="server" Text="* Object Id:" />
        <ektronUI:TextField ID="uxObjectId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"/>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTagTypeLabel" AssociatedControlID="uxTagType"
CssClass="span-4 last" runat="server" Text="* Tag Type:" />
        <asp:DropDownList ID="uxTagType" runat="server">
            <asp:ListItem Value="0">Content</asp:ListItem>
            <asp:ListItem Value="1">Folder</asp:ListItem>
            <asp:ListItem Value="2">User</asp:ListItem>
            <asp:ListItem Value="3">UserGroup</asp:ListItem>
            <asp:ListItem Value="4">Library</asp:ListItem>
            <asp:ListItem Value="5">Collection</asp:ListItem>
        </asp:DropDownList>
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long objectId = long.Parse(uxObjectId.Text);
            EkEnumeration.CMSObjectTypes cmsObjectTypes =
(EkEnumeration.CMSObjectTypes)Convert.ToInt32(uxTagType.SelectedItem.Value);

            TagManager tagManager = new TagManager();
            tagManager.Tag(uxTagText.Text, objectId, cmsObjectTypes);

            MessageUtilities.UpdateMessage(uxMessage, "Tagging successful",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.TagData)
```

Updates an existing TagData item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Tag Type

Parameters

- `tagData`. The TagData object to update.

Returns

Returns the custom TagData object update

Remarks

Validate the TagData item with `GetItem()` before you update the properties. Then, call `Update` with your modified TagData object.

NOTE: You cannot change all properties after the initial Add event, such as the TagData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTagIdLabel" AssociatedControlID="uxTagId" CssClass="span-4
last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTagId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"/>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTagTypeLabel" AssociatedControlID="uxTagType"
CssClass="span-4 last" runat="server" Text="* Tag Type:" />
        <asp:DropDownList ID="uxTagType" runat="server">
            <asp:ListItem>ContentDefault</asp:ListItem>
            <asp:ListItem>UserDefault</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long tagId = long.Parse(uxTagId.Text);

            TagManager tagManager = new TagManager();
            TagData tagData = tagManager.GetItem(tagId);

            if (tagData != null)
            {
                tagData.Type = uxTagType.Text == "ContentDefault" ?
TagTypes.ContentDefault : TagTypes.UserDefault;

                tagManager.Update(tagData);

                MessageUtilities.UpdateMessage(uxMessage,"Tag updated ",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Tag does not exist. ",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

TagCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Community
```

Constructors

- `TagCriteria()`

  ```
  public TagCriteria()
  ```

- `TagCriteria(Ektron.Cms.Common.TagProperty, EkEnumeration.OrderByDirection)`

  ```
  public TagCriteria(Ektron.Cms.Common.TagProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `TagProperty` are:

  - Id
  - LanguageId
  - Text
  - Type

TagData

Namespace

```
Ektron.Cms
```

Properties

- `Id`. ID of tag, which should be set only for existing tags. New tags will be given an ID when they are saved.

  ```
  public long Id { set; get; }
  ```

- `LanguageId`. Language of the tag.

  ```
  public int LanguageId { set; get; }
  ```

- `TagData()`. Constructor.

  ```
  public TagData()
  ```

- `TagData(string)`. Constructor.

  ```
  public TagData(string text)
  ```

- `TagData(string, int)`. Constructor.

  ```
  public TagData(string text, int languageId)
  ```

- `TagData(sting, int, long)`. Constructor.

  ```
  public TagData(string text, int languageId, long id)
  ```

- `Text`. The tag's actual text.

  ```
  public string Text { set; get; }
  ```

- `TotalUsedCount`. Returns the total number of times this tag has been used on an object.

  ```
  public int TotalUsedCount { set; get; }
  ```

- `Type`. The type of tag.

```
public Ektron.Cms.TagTypes Type { set; get; }
```

- `Validate()`

```
public override ValidationResults Validate()
```

Content

8.50 and higher

The Content manager category manages Web content and has the following classes:

- *AssetManager* on the facing page. Manages assets, which are files that are created outside of Ektron CMS.
- *ContentManager* on page 1076. Manages content (including html, text, documents, and digital media).
- *ContentRatingManager* on page 1124. Manages user valuations on content.
- *LibraryManager* on page 1142. Manages libraries, which stores images, files, quicklinks, and hyperlinks that can be inserted into editor content.
- *MetadataTypeManager* on page 1157. Manages metadata types, such as title and other artifacts.
- *TemplateManager* on page 1172. Manages templates, which typically includes page headers, footers, and placeholders for content, forms, summaries, calendars, collections, or other page elements.

AssetManager

8.50 and higher

The AssetManager class manages assets, which are files that are created outside of Ektron CMS.

Namespace

```
Ektron.Cms.Framework.Content.AssetManager
```

Constructors

- `AssetManager()`
- `AssetManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `AssetService`. Returns an instance of the business objects Asset Service.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.ContentAssetData)
```

Adds an asset.

NOTE: This method is not supported in WCF mode.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID
- * Title
- * File Path
- Keywords
- Summary

Parameters

- `assetData`. The ContentAssetData object.

Returns

Returns the custom ContentAssetData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last" runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxContentTitleLabel" AssociatedControlID="uxContentTitle"
CssClass="span-3 last" runat="server" Text="* Title:" />
        <ektronUI:TextField  ID="uxContentTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxFilePathLabel" AssociatedControlID="uxFilePath" CssClass="span-
3 last" runat="server" Text="* File Path:" />
        <asp:FileUpload ID="uxFilePath" size="29.75" CssClass="span-4" runat="server" />
    </li>

    <li class="clearfix">
        <asp:Label ID="uxContentKeywordsLabel" AssociatedControlID="uxContentKeywords"
CssClass="span-3 last" runat="server" Text="Keywords(;):" />
        <ektronUI:TextField  ID="uxContentKeywords" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <asp:Label ID="uxContentSummaryLabel" AssociatedControlID="uxContentSummary"
CssClass="span-3 last" runat="server" Text="Summary:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxContentSummary"
CssClass="span-4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Content;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AssetManager assetManager = new AssetManager();
            HttpPostedFile postedFile = uxFilePath.PostedFile;
            int fileLength = postedFile.ContentLength;
            byte[] fileData = new byte[fileLength];
            postedFile.InputStream.Read(fileData, 0, fileLength);
            ContentAssetData contentAssetData = new ContentAssetData()
            {
                FolderId = long.Parse(uxFolderId.Text),
                Title = uxContentTitle.Text,
                Teaser = uxContentSummary.Text,
                File = fileData,
                LanguageId = assetManager.RequestInformation.ContentLanguage,
                AssetData = new Ektron.Cms.Common.AssetData()
                {
                    FileName = Path.GetFileName(postedFile.FileName)
                }
            };

            contentAssetData.MetaData = new ContentMetaData[1];
            contentAssetData.MetaData[0] = new ContentMetaData()
```

```
            {
                Id = 116,
                Text = uxContentKeywords.Text
            };

            assetManager.Add(contentAssetData);
            MessageUtilities.UpdateMessage(uxMessage, "Content Added with Id " +
contentAssetData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes an asset from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `contentId`. The ID of the asset to be deleted.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
```

```
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long contentID = long.Parse(uxContentId.Text);

            AssetManager assetManager = new AssetManager();
            assetManager.Delete(contentID);

            MessageUtilities.UpdateMessage(uxMessage,  "Content with Id " +
uxContentId.Text + " deleted", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific ContentData object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- Return Metadata

Parameters

- `id`. ID of the content to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>

    <li class="clearfix">
        <asp:Label ID="uxReturnMetadataLabel" AssociatedControlID="uxReturnMetadata"
CssClass="span-3 last" runat="server" Text="Return Metadata:" />
        <ektronUI:Button ID="uxReturnMetadata" DisplayMode="Checkbox" Text=""
CssClass="span-4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
    </ol>

    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLanguageID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxMetadata" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Content;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long contentID = long.Parse(uxContentId.Text);
            bool returnMetadata = uxReturnMetadata.Checked;

            AssetManager assetManager = new AssetManager();
            ContentAssetData assetData = assetManager.GetItem(contentID,returnMetadata);

            if (assetData != null)
            {

                MessageUtilities.UpdateMessage(uxMessage, "Details for  Content Id " +
assetData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxTitle.Text  =  "Title : " + assetData.Title;
                uxLanguageID.Text = "Language ID : " + assetData.LanguageId;
                uxContentType.Text = "Content Type : " + assetData.ContType ;
                uxMetadata.Text += "Metadata : ";
                if (assetData.MetaData != null)
                {
                    foreach (ContentMetaData cMeta in assetData.MetaData)
                    {
                        uxMetadata.Text += cMeta.Name  + ":" + cMeta.Text + "; ";
                    }
                }
                else
                {
                    uxMetadata.Text += "null";
                }
            }
            else
            {
                 MessageUtilities.UpdateMessage(uxMessage, "Either content with ID " +
contentID + " does not exist or is not an Asset ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
```

```
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Content.AssetCriteria)
```

Retrieves lists of AssetCriteria objects.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Asset Property
- * Object Value

Parameters

- `criteria`. Criteria used to retrieve content.

Returns

List of asset objects matching the supplied criteria.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAssetPropertyLabel" AssociatedControlID="uxAssetProperty"
CssClass="span-4 last" runat="server" Text="* AssetProperty:" />
        <asp:DropDownList ID="uxAssetProperty" runat="server">
            <asp:ListItem>Title</asp:ListItem>
            <asp:ListItem>FolderId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
```

```
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxContentAssetDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                        <th>
                        Title
                    </th>
                    <th>
                        Folder Id
                    </th>
                    <th>
                        Content Type
                    </th>
                    <th>
                        Date Created
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FolderId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ContType")%>
            </td>
            <td class="devsite-method">
                <%# Eval("DateCreated")%>
            </td>
        </tr>
    </ItemTemplate>
```

```
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Content;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            AssetManager assetManager = new AssetManager();

            AssetCriteria criteria = new AssetCriteria();
            if (uxAssetProperty.SelectedItem.Text == "FolderId")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(AssetProperty.FolderId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(AssetProperty.Title, CriteriaFilterOperator.Contains,
Objectvalue);
            }

            List<ContentAssetData> contentAssetDataList = assetManager.GetList
(criteria);

            uxContentAssetDataListView.Visible = true;
            uxContentAssetDataListView.DataSource = contentAssetDataList;
            uxContentAssetDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.ContentAssetData)
```

Updates an existing ContentAssetData item in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Folder ID
- * Content ID
- Title
- File Path
- Keywords
- Summary

Parameters

- `assetData`. The asset data object.

Returns

Returns the custom ContentAssetData object updated.

Remarks

Validate the asset item with `GetItem()` before you update the properties. Then, call `Update` with your modified ContentAssetData object.

NOTE: You cannot change all properties after the initial Add event, such as the ContentAssetData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last" runat="server" Text="Folder Id:" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <asp:Label ID="uxContentTitleLabel" AssociatedControlID="uxContentTitle"
CssClass="span-3 last" runat="server" Text="Title:" />
        <ektronUI:TextField  ID="uxContentTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxFilePathLabel" AssociatedControlID="uxFilePath" CssClass="span-
3 last" runat="server" Text="File Path:" />
        <asp:FileUpload ID="uxFilePath" size="29.75" CssClass="span-4" runat="server" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxContentKeywordsLabel" AssociatedControlID="uxContentKeywords"
CssClass="span-3 last" runat="server" Text="Keywords:" />
        <ektronUI:TextField  ID="uxContentKeywords" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxContentSummaryLabel" AssociatedControlID="uxContentSummary"
CssClass="span-3 last" runat="server" Text="Summary:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxContentSummary"
CssClass="span-4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
            <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                Text="Status:" />
        </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            long contentID = long.Parse(uxContentId.Text);
            AssetManager assetManager = new AssetManager();

            HttpPostedFile postedFile = uxFilePath.PostedFile;

            int fileLength = postedFile.ContentLength;
            byte[] fileData = new byte[fileLength];
            postedFile.InputStream.Read(fileData, 0, fileLength);


            ContentAssetData contentAssetData = assetManager.GetItem(contentID);
            if (contentAssetData != null)
            {
                contentAssetData.Title = uxContentTitle.Text != string.Empty ?
uxContentTitle.Text : contentAssetData.Title;
                contentAssetData.Teaser = uxContentSummary.Text != string.Empty ?
uxContentSummary.Text : contentAssetData.Teaser;
                if (!string.IsNullOrEmpty(Path.GetFileName(postedFile.FileName)))
                {
                    contentAssetData.File = fileData;
                    contentAssetData.AssetData.FileName = Path.GetFileName
(postedFile.FileName);
                }
                if (!string.IsNullOrEmpty(uxContentKeywords.Text))
                {
                    contentAssetData.MetaData = new ContentMetaData[1];
                    contentAssetData.MetaData[0] = new ContentMetaData()
                    {
                        Id = 116,
                        Text = uxContentKeywords.Text
                    };
                }
                assetManager.Update(contentAssetData);

                MessageUtilities.UpdateMessage(uxMessage, "Content Updated with Id " +
uxContentId.Text, Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid DMS Content
ID.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

AssetCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Content
```

Constructors

- `AssetCriteria()`

  ```
  public ContentAssetCriteria()
  ```

- `AssetCriteria(Ektron.Cms.Content.AssetProperty, EkEnumeration.OrderByDirection)`

  ```
  public ContentAssetCriteria(Ektron.Cms.Content.AssetProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `AssetProperty` are:

  - AssetId
  - DateCreated
  - DateModified
  - FolderId
  - FolderName
  - Id
  - IsArchived
  - IsPublished
  - LanguageId
  - LastEditorFirstName
  - LastEditorLastName
  - Status
  - SubType
  - Title
  - Type
  - UserId
  - XmlConfigurationId

ContentAssetData

Namespace

```
Ektron.Cms.Content
```

Properties

- `AssetId`. Gets or sets the Asset ID of the content item for the AssetInfoData object and returns an Asset ID of the content item.

```
public string AssetId { set; get; }
```

- `File`. Gets or Sets an array of data that represents the asset's contents.

```
public byte[] File { set; get; }
```

- `FlagDefId`. Not applicable to this class..

```
public new long FlagDefId { set; get; }
```

- `Html`. Not applicable to this class.

```
public new string Html { set; get; }
```

- `Hyperlink`. Not applicable to this class.

```
public new string HyperLink { set; get; }
```

- `IsXmlInherited`. Not applicable to this class.

```
public new bool IsXmlInherited { set; get; }
```

- `LegacyData`. Not applicable to this class.

```
public new object LegacyData { set; get; }
```

- `MediaText`. Not applicable to this class.

```
public new string MediaText { set; get; }
```

- `StyleSheet`. Not applicable to this class.

```
public new string StyleSheet { set; get; }
```

- `TemplateConfiguration`. Not applicable to this class.

```
public new string TemplateConfiguration { set; get; }
```

- `Validate()`. Validates this instance and returns a ValidationResults object.

```
public override ValidationResults Validate()
```

- `XmlConfiguration`. Not applicable to this class.

```
public new XmlConfigData XmlConfiguration { set; get; }
```

- `XmlInheritedFrom`. Not applicable to this class.

```
public new long XmlInheritedFrom { set; get; }
```

ContentManager

8.50 and higher

The ContentManager class manages content (including HTML, text, documents, and digital media).

Namespace

```
Ektron.Cms.Framework.Content.ContentManager
```

Constructors

- `ContentManager()`
- `ContentManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `ContentService`. Returns an instance of the business objects Content Service.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* on the facing page
- `Approve(Int64)`. Submits a content item for publishing.
- *AssignTaxonomy* on page 1080
- *Cancel* on page 1082
- *CheckIn* on page 1083
- *CheckOut* on page 1085
- *CopyContent* on page 1086
- *Delete* on page 1088

- `GetAssignedTaxonomyList(Int64,Int32)`. 8.70 and higher Retrieves a list of category or taxonomy data to which a given content is assigned.
- `GetContentByHistoryId(Int64)`. 8.60 and higher Retrieves the corresponding content by the history ID.
- `GetHistoryList(Int64)`. 8.60 and higher Retrieves a list of ContentHistoryData objects based on Content ID.
- `GetHtmlDifference(String,String)`. 8.60 and higher Returns the differences between 2 strings of content HTML.
- *GetItem* on page 1090
- *GetList (content collection criteria)* on page 1092
- *GetList (content criteria)* on page 1095
- *GetList (metadata criteria)* on page 1098
- *GetList (content taxonomy)* on page 1101
- *MoveContent* on page 1104
- *RemoveTaxonomy* on page 1106
- `Save(ContentData)`. 8.60 and higher Saves an existing content item in the CMS.
- *Submit* on page 1108
- *Update* on page 1110

Add

```
Add(Ektron.Cms.ContentData)
```

Adds content to the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID
- * Title
- Keywords (separate with a semicolon (;))
- Summary
- Private
- * HTML

Parameters

- `contentData`. The ContentData object to add to the CMS.

Returns

Returns the custom ContentData object added.

Remarks

When you add the content, the method returns a new Content ID. When you add content using this method, Web alerts are not sent.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last" runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxContentTitleLabel" AssociatedControlID="uxContentTitle"
CssClass="span-3 last" runat="server" Text="* Title:" />
        <ektronUI:TextField  ID="uxContentTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxContentKeywordsLabel" AssociatedControlID="uxContentKeywords"
CssClass="span-3 last" runat="server" Text="Keywords(;):" />
        <ektronUI:TextField  ID="uxContentKeywords" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxContentSummaryLabel" AssociatedControlID="uxContentSummary"
CssClass="span-3 last" runat="server" Text="Summary:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxContentSummary"
CssClass="span-4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxPrivateLabel" AssociatedControlID="uxPrivate" CssClass="span-3
last" runat="server" Text=" Private:" />
        <ektronUI:Button DisplayMode="Checkbox"  ID="uxPrivate" Text="" CssClass="span-
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxContentHtmlLabel" AssociatedControlID="uxContentHtml"
CssClass="span-3 last" runat="server" Text="* Html:" />
        <asp:TextBox TextMode="MultiLine" Rows="4"  ID="uxContentHtml" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
            <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
            <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server"
```

```
Text="* - Required" />
        </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ContentManager contentManager = new ContentManager();
            ContentData contentData = new ContentData()
            {
                FolderId = long.Parse(uxFolderId.Text),
                Title = uxContentTitle.Text,
                Teaser = uxContentSummary.Text,
                Html = uxContentHtml.Text,
                IsPrivate = uxPrivate.Checked,
                IsPermissionsInherited = true
            };

            contentManager.Add(contentData);

            MessageUtilities.UpdateMessage(uxMessage, "Content Saved with Id " +
contentData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

            contentManager.AssignTaxonomy(contentData.Id, 68);
            contentManager.AssignTaxonomy(contentData.Id, 189);
            contentManager.AssignTaxonomy(contentData.Id, 190);
            contentManager.AssignTaxonomy(contentData.Id, 192);

            UserPermissionData permission = new UserPermissionData()
            {
                CanAdd = true,
                CanApprove = true,
                CanDelete = true,
                CanEdit = true,
                CanPublish = true,
                IsAdmin = true,
                ContentId = contentData.Id,
                UserId = 8,
```

```
                FolderId = contentData.FolderId,
                IsInherited = true
            };

            Ektron.Cms.Framework.Settings.PermissionManager perManager = new
Ektron.Cms.Framework.Settings.PermissionManager
(Ektron.Cms.Framework.ApiAccessMode.Admin);
            perManager.Add(permission);


        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

AssignTaxonomy

```
AssignTaxonomy(System.Int64,System.Int64)
```

Assigns a content item from a taxonomy category.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID
- * Taxonomy ID

Parameters

- `contentId`. ID of the content item to assign to a category.
- `categoryId`. ID of the taxonomy category to assign content to.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIDLabel" AssociatedControlID="uxTaxonomyID"
CssClass="span-3 last" runat="server" Text="* Taxonomy Id:" />
```

```
        <ektronUI:TextField ID="uxTaxonomyID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentStatus" Visible="false"
AssociatedControlID="uxContentStatus" CssClass="span-3 last" runat="server"
Text="Status:" />
    </li>

                    <li class="clearfix" style="color: red;">
                       <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9
last" runat="server"
                            Text="Status:" />
                    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Assign"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ContentManager contentManager = new ContentManager();
            ContentData cData = new ContentData();
            long contentID = long.Parse(uxContentId.Text);
            long TaxonomyID = long.Parse(uxTaxonomyID.Text);
            bool result = Information.IsNumeric(uxContentId.Text);
            cData = contentManager.GetItem(contentID, false);
            if (!((result) && cData != null))
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Invalid Content ID",
Message.DisplayModes.Error);
                return;
            }
            TaxonomyManager taxonomyManager = new TaxonomyManager();
            TaxonomyData taxonomyData = taxonomyManager.GetItem(TaxonomyID);
```

```
            if (taxonomyData != null && taxonomyData.Id > 0)
            {
                //Assign content taxonomy to a category
                contentManager.AssignTaxonomy(contentID, TaxonomyID);
                MessageUtilities.UpdateMessage(uxMessage,  " Content Id : " + contentID
+ " has been Assigned to Taxonomy ID : " + TaxonomyID, Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Taxonomy
Id.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Cancel

8.60 and higher

```
Cancel(System.Int64)
```

Undo the last changes of the content.

Authenticated Users

- CMS Administrators

Fields

*=Required

- * Content ID

Parameters

- `contentId`. ID of the content item.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Cancel"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using Ektron.Cms.Framework.Content;
using Microsoft.VisualBasic;
using Ektron.Cms;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ContentManager contentManager = new ContentManager();
            long contentID = long.Parse(uxContentId.Text);
            contentManager.Cancel(contentID);
            MessageUtilities.UpdateMessage(uxMessage, contentID + " content undo the
last changes", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

CheckIn

```
CheckIn(System.Int64)
```

Checks in a content item to the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID

Parameters

- `contentId`. ID of the content item to be checked in.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentStatus" Visible="false"
AssociatedControlID="uxContentStatus" CssClass="span-3 last" runat="server"
Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Check In"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
using Microsoft.VisualBasic;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ContentManager contentManager = new ContentManager();
            ContentData cData = new ContentData();
            long contentID = long.Parse(uxContentId.Text);
            bool result = Information.IsNumeric(uxContentId.Text);
            cData = contentManager.GetItem(contentID, false);
            if (!((result) && cData != null && cData.IsPublished))
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Invalid Content ID",
Message.DisplayModes.Error);
```

```
                return;
            }
            contentManager.CheckIn(contentID);
            MessageUtilities.UpdateMessage(uxMessage,  "Content successfully checked-in
and  its content id is : " + contentID, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

CheckOut

```
CheckOut(System.Int64)
```

Checks out a content item from the CMS.

Authenticated Users

- CMS Administrators

NOTE: Any user can use the Update method to update content on folders to which the user has permission.

Fields

*=Required

- * Content ID

Parameters

- `contentId`. ID of the content item to be checked out.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:TextField ID="uxContentStatus" Visible="false"
AssociatedControlID="uxContentStatus" CssClass="span-3 last" runat="server"
Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Check Out"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
```

```
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
using Microsoft.VisualBasic;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ContentManager contentManager = new ContentManager();
            ContentData cData = new ContentData();
            long contentID = long.Parse(uxContentId.Text);
            bool result = Information.IsNumeric(uxContentId.Text);
            cData = contentManager.GetItem(contentID, false);
            if (!((result) && cData != null && cData.IsPublished))
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Invalid Content ID",
Message.DisplayModes.Error);
                return;
            }
            contentManager.CheckOut(contentID);
            MessageUtilities.UpdateMessage(uxMessage,"Content successfully checked-out
and  its content id is : " + contentID, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

CopyContent

8.60 and higher

```
CopyContent(System.Int64,System.Int64,System.Int64,System.Boolean)
```

Copy content from one folder to another folder that specified as a folder ID.

Authenticated Users

- CMS Administrators

Fields

*=Required

- * Content ID
- * Folder ID
- * Language ID
- * Is Published

Parameters

- `contentId`. ID of the content item to be copied.
- `folderId`. ID of the folder into which the content is copied.
- `languageId`. ID of the language of the content item.
- `isPublish`. Determines whether the content is to be published.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIDLabel" AssociatedControlID="uxFolderID"
CssClass="span-3 last" runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="uxFolderID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageIDLabel" AssociatedControlID="uxLanguageID"
CssClass="span-3 last" runat="server" Text="* Language Id:" />
        <ektronUI:TextField ID="uxLanguageID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1033" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsPublishLabel" AssociatedControlID="uxIsPublish"
CssClass="span-3 last" runat="server" Text="* IsPublish:" />
        <ektronUI:TextField ID="uxIsPublish" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="True" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentStatus" Visible="false"
AssociatedControlID="uxContentStatus" CssClass="span-3 last" runat="server"
Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
```

```
Text="Copy"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            long folderId,contentId,languageId;
            contentId=Convert.ToInt64(uxContentId.Text);
            languageId=Convert.ToInt64(uxLanguageID.Text);
            bool publish=Convert.ToBoolean(uxIsPublish.Text);
            folderId = Convert.ToInt64(uxFolderID.Text);
            ContentManager contentManager = new ContentManager();
            contentManager.ContentLanguage = (int)languageId;
            contentManager.CopyContent(contentId, folderId, languageId, publish);
            MessageUtilities.UpdateMessage(uxMessage, contentId+" Content copy to folder
" + folderId.ToString(), Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);


        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a content item from the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID

Parameters

- `contentId`. ID of the content item to be deleted.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentStatus"  Visible="false"
AssociatedControlID="uxContentStatus" CssClass="span-3 last" runat="server"
Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
using Microsoft.VisualBasic;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ContentManager contentManager = new ContentManager();
            ContentData cData = new ContentData();
            long contentID = long.Parse(uxContentId.Text);
            bool result = Information.IsNumeric(uxContentId.Text);
            cData = contentManager.GetItem(contentID, false);
            if (!((result) && cData != null && cData.IsPublished))
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus,  "Invalid Content ID",
Message.DisplayModes.Error);
                return;
            }
            contentManager.Delete(contentID);
            MessageUtilities.UpdateMessage(uxMessage, "Content successfully deleted and
its content id was : " + contentID, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64,System.Boolean)
```

Retrieves the properties of a specific ContentData object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID

Parameters

- `id`. ID of the content to retrieve.
- `returnMetadata`. Flag indicating whether or not ContentMetadata should also be retrieved with the content item.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentStatus" Visible="false" CssClass="span-3 last"
runat="server" Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetDetails"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentLanguage" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
using Microsoft.VisualBasic;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ContentManager contentManager = new ContentManager();
            ContentData cData = new ContentData();
```

```
            Boolean returnMetadata = false;
            long contentID = long.Parse(uxContentId.Text);
            bool result = Information.IsNumeric(uxContentId.Text);
            cData = contentManager.GetItem(contentID,returnMetadata);
            if (!((result) && cData != null))
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Invalid content ID ",
Message.DisplayModes.Error);
                return;
            }
            uxContID.Text = "Content ID : " + cData.Id;
            uxContentTitle.Text = "Content Title : " + cData.Title;
            uxContentLanguage.Text = "ContentLanguage Id : " + cData.LanguageId;
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (content collection criteria)

```
GetList(Ektron.Cms.Content.ContentCollectionCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Collection Property
- * Object Value
- In Preview Mode

Parameters

- `criteria`. ContentCollectionCriteria used to retrieve content by collections.

Returns

List of ContentData objects matching the supplied criteria.

See *Criteria use for GetList methods* on page 156 for additional information.

NOTE: If the content you request does not exist in the current language, content is returned in fallback languages according to the fallback locale chain. However, if you request content in a specific language using a `CriteriaFilter`, the fallback languages are not returned because you are overriding the default language functionality and looking for specific content. For information about setting the fallback locale, see Setting the Fallback Locale.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentPropertyLabel"
AssociatedControlID="uxContentProperty" CssClass="span-4 last"
            runat="server" Text="* Collection Property :" />
        <asp:DropDownList ID="uxContentProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>

        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxInPreviewModeLabel" AssociatedControlID="uxInPreviewMode"
CssClass="span-4 last"
            runat="server" Text=" InPreviewMode :" />
        <asp:CheckBox ID="uxInPreviewMode" runat="server" Checked="false" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxContentlist" runat="server" ItemPlaceholderID="aspItemPlaceholder"
Visible="false">
    <EmptyDataTemplate>Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Content Id
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
```

```
                        Status
                    </th>
                    <th>
                        Date Modified
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Status")%>
            </td>
            <td class="devsite-method">
                <%# Eval("DateModified")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Content;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            string Objectvalue;
            string Property = uxContentProperty.SelectedItem.Text;

            ContentManager contentManager = new ContentManager();
```

```
            if(uxInPreviewMode.Checked)
                contentManager.InPreviewMode = uxInPreviewMode.Checked;

            ContentCollectionCriteria criteria = new ContentCollectionCriteria
(ContentProperty.Id, EkEnumeration.OrderByDirection.Ascending);

            Objectvalue = uxObjectValue.Text;

            if (Property == "Id")
            {
                criteria.AddFilter(Convert.ToInt64(uxObjectValue.Text));
            }
            else
            {
                criteria.AddFilter(Objectvalue);
            }

            List<ContentData> contentList = contentManager.GetList(criteria);

            uxContentlist.Visible = true;
            uxContentlist.DataSource = contentList;
            uxContentlist.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (content criteria)

```
GetList(Ektron.Cms.Content.ContentCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Collection Property
- * Object Value

Parameters

- `criteria`. ContentCriteria used to retrieve content.

Returns

List of ContentData objects matching the supplied criteria.

See *Criteria use for GetList methods* on page 156 for additional information.

NOTE: If the content you request does not exist in the current language, content is returned in fallback languages according to the fallback locale chain. However, if you request content in a specific language using a `CriteriaFilter`, the fallback languages are not returned because you are overriding the default language functionality and looking for specific content. For information about setting the fallback locale, see Setting the Fallback Locale in the Ektron Reference.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentPropertyLabel"
AssociatedControlID="uxContentProperty" CssClass="span-4 last"
            runat="server" Text="* Collection Property :" />
        <asp:DropDownList ID="uxContentProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>

        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxInPreviewModeLabel" AssociatedControlID="uxInPreviewMode"
CssClass="span-4 last"
            runat="server" Text=" InPreviewMode :" />
        <asp:CheckBox ID="uxInPreviewMode" runat="server" Checked="false" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxContentlist" runat="server" ItemPlaceholderID="aspItemPlaceholder"
Visible="false">
    <EmptyDataTemplate>Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
```

```
                        Content Id
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
                        Status
                    </th>
                    <th>
                        Date Modified
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Status")%>
            </td>
            <td class="devsite-method">
                <%# Eval("DateModified")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Content;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
```

```
        {
            object Objectvalue;
            string Property = uxContentProperty.SelectedItem.Text;

            ContentManager contentManager = new ContentManager();
            ContentCriteria criteria = new ContentCriteria(ContentProperty.Id,
EkEnumeration.OrderByDirection.Ascending);

            Objectvalue = Convert.ToInt64(uxObjectValue.Text);

            if (Property == "Id")
            {
                criteria.AddFilter(ContentProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else if (Property == "FolderID")
            {
                criteria.AddFilter(ContentProperty.FolderId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                criteria.AddFilter(ContentProperty.LanguageId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            List<ContentData> contentList = contentManager.GetList(criteria);

            uxContentlist.Visible = true;
            uxContentlist.DataSource = contentList;
            uxContentlist.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (metadata criteria)

```
GetList(Ektron.Cms.Content.ContentMetadataCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content Property
- * Metadata ID
- * Value

Parameters

- `criteria`. ContentMetadataCriteria used to retrieve content.

Returns

List of ContentData objects matching the supplied criteria.

See *Criteria use for GetList methods* on page 156 for additional information.

NOTE: If the content you request does not exist in the current language, content is returned in fallback languages according to the fallback locale chain. However, if you request content in a specific language using a `CriteriaFilter`, the fallback languages are not returned because you are overriding the default language functionality and looking for specific content. For information about setting the fallback locale, see Setting the Fallback Locale in the Ektron Reference.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentPropertyLabel"
AssociatedControlID="uxContentProperty" CssClass="span-4 last" runat="server" Text="*
ContentProperty :" />
        <asp:DropDownList ID="uxContentProperty" runat="server" >
            <asp:ListItem>Metadata ID</asp:ListItem>

        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMetadataIDLabel" AssociatedControlID="uxMetadataID"
CssClass="span-4 last" runat="server" Text="* Metadata ID :" />
        <ektronUI:TextField ID="uxMetadataID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="117" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMetaValueLabel" AssociatedControlID="uxMetaValue"
CssClass="span-4 last" runat="server" Text="* Value :" />
        <ektronUI:TextField ID="uxMetaValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="SampleMeta" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxContentlist" runat="server" ItemPlaceholderID="aspItemPlaceholder"
Visible="false">
    <EmptyDataTemplate>Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        ID
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
                        LanguageId
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("ID")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("LanguageId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Content;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            string Property = uxContentProperty.SelectedItem.Text;
            ContentManager contentManager = new ContentManager();
            ContentMetadataCriteria criteria = new ContentMetadataCriteria
(ContentProperty.Id, EkEnumeration.OrderByDirection.Ascending);


            Objectvalue = Convert.ToString(uxMetaValue.Text);
            criteria.AddFilter(long.Parse(uxMetadataID.Text),
CriteriaFilterOperator.EqualTo, Objectvalue);

            List<ContentData> contentList = contentManager.GetList(criteria);

            uxContentlist.Visible = true;
            uxContentlist.DataSource = contentList;
            uxContentlist.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (content taxonomy)

```
GetList(Ektron.Cms.Content.ContentTaxonomyCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content Property
- * Taxonomy ID
- * Recursive

Parameters

- `criteria`. ContentTaxonomyCriteria used to retrieve content.

Returns

List of ContentData objects matching the supplied criteria.

See *Criteria use for GetList methods* on page 156 for additional information.

NOTE: If the content you request does not exist in the current language, content is returned in fallback languages according to the fallback locale chain. However, if you request content in a specific language using a `CriteriaFilter`, the fallback languages are not returned because you are overriding the default language functionality and looking for specific content. For information about setting the fallback locale, see Setting the Fallback Locale in the Ektron Reference.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentTaxonomyPropertyLabel"
AssociatedControlID="uxContenttaxonomyProperty" CssClass="span-4 last" runat="server"
Text="* ContentProperty :" />
        <asp:DropDownList ID="uxContentTaxonomyProperty" runat="server"
AutoPostBack="true" OnSelectedIndexChanged="uxContentTaxonomyProperty_
SelectedIndexChanged">
            <asp:ListItem>Taxonomy ID</asp:ListItem>
            <asp:ListItem>Taxonomy Path</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIDLabel" AssociatedControlID="uxTaxonomyID"
CssClass="span-4 last" runat="server" Text="* Taxonomy ID :" />
        <ektronUI:TextField ID="uxTaxonomyID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="189" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyPathLabel" Visible="false"
AssociatedControlID="uxTaxonomyPath" CssClass="span-4 last" runat="server" Text="*
Taxonomy Path(start with  '\' and separate by '\'):"  />
        <ektronUI:TextField ID="uxTaxonomyPath" Visible="false" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup"  Text="\OnTrek Site Navigation"
/>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxRecursiveLabel"
AssociatedControlID="uxTaxRecursiveValue" CssClass="span-4 last" runat="server" Text="*
Recursive :" />
        <asp:DropDownList ID="uxTaxRecursiveValue" runat="server">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
```

```
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxContentlist" runat="server" ItemPlaceholderID="aspItemPlaceholder"
Visible="false">
    <EmptyDataTemplate>Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Content ID
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
                        LanguageId
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("ID")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("LanguageId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
```

```
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Content;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            bool recursive =  Convert.ToBoolean(uxTaxRecursiveValue.SelectedValue);
            string Property = uxContentTaxonomyProperty.SelectedItem.Text;
            ContentManager contentManager = new ContentManager();
            ContentTaxonomyCriteria criteria = new ContentTaxonomyCriteria
(ContentProperty.Id, EkEnumeration.OrderByDirection.Ascending);

            if (Property == "Taxonomy ID")
            {

                criteria.AddFilter(long.Parse(uxTaxonomyID.Text), recursive);
            }
            else if (Property == "Taxonomy Path")
            {

                criteria.AddFilter(uxTaxonomyPath.Text, recursive);
            }
            List<ContentData> contentList = contentManager.GetList(criteria);

            uxContentlist.Visible = true;
            uxContentlist.DataSource = contentList;
            uxContentlist.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

MoveContent

8.60 and higher

```
MoveContent(System.Collections.Generic.List,System.Int64)
```

Move content from one folder to another folder that specified as a folder ID.

Authenticated Users

- CMS Administrators

Fields

*=Required

- * Content IDs
- * Folder ID

Parameters

- `contentIds`. ID of the content items to be move.
- `folderId`. ID of the folder into which the content is moved.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Ids:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIDLabel" AssociatedControlID="uxFolderID"
CssClass="span-3 last" runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="uxFolderID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentStatus" Visible="false"
AssociatedControlID="uxContentStatus" CssClass="span-3 last" runat="server"
Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Move"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            String[] idList;

            List<long> Ids = new List<long>();
```

```
            long folderId;
            string idStr;
            idStr = uxContentId.Text.Trim();
            idList = idStr.Split(',');

            foreach (string id in idList)
            {
                Ids.Add(Convert.ToInt64(id.Trim()));
            }

            folderId = Convert.ToInt64(uxFolderID.Text);
            ContentManager contentManager = new ContentManager();
            contentManager.MoveContent(Ids, folderId);
            MessageUtilities.UpdateMessage(uxMessage, "Content moved to folder " +
folderId.ToString(), Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);



          }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

RemoveTaxonomy

```
RemoveTaxonomy(System.Int64,System.Int64)
```

Removes a content item from a taxonomy category.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID
- * Taxonomy ID

Parameters

- `contentId`. ID of the content item to remove from category.
- `categoryId`. ID of the taxonomy category to remove content from.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIDLabel" AssociatedControlID="uxTaxonomyID"
CssClass="span-3 last" runat="server" Text="* Taxonomy Id:" />
        <ektronUI:TextField ID="uxTaxonomyID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentStatus" Visible="false"
AssociatedControlID="uxContentStatus" CssClass="span-3 last" runat="server"
Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Remove"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ContentManager contentManager = new ContentManager();
            ContentData cData = new ContentData();
            long contentID = long.Parse(uxContentId.Text);
            long TaxonomyID = long.Parse(uxTaxonomyID.Text);
            bool result = Information.IsNumeric(uxContentId.Text);
            cData = contentManager.GetItem(contentID, false);
            if (!((result) && cData != null))
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Invalid Content ID",
```

```
Message.DisplayModes.Error);
                return;
            }
            //Remove content taxonomy to a category
            contentManager.RemoveTaxonomy(contentID, TaxonomyID);
            MessageUtilities.UpdateMessage(uxMessage, " Content Id : " + contentID + "
has been removed to Taxonomy ID : " + TaxonomyID, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Submit

```
Submit(System.Int64)
```

Submits a content item to be published.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID

Parameters

- `contentId`. ID of the content item to be submitted.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentStatus" Visible="false"
AssociatedControlID="uxContentStatus" CssClass="span-3 last" runat="server"
Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
```

```
Text="Submit"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
using Microsoft.VisualBasic;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            ContentManager contentManager = new ContentManager();
            ContentData cData = new ContentData();
            long contentID = long.Parse(uxContentId.Text);
            bool result = Information.IsNumeric(uxContentId.Text);
            if (!((result)))
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Invalid Content ID",
Message.DisplayModes.Error);
                return;
            }
            contentManager.Submit(contentID);
            cData = contentManager.GetItem(contentID, false);
            if (cData.Status == "S")
            {
                MessageUtilities.UpdateMessage(uxMessage, "Content successfully
submitted and  its content id is : " + contentID + "and content Status is :" +
cData.Status, Message.DisplayModes.Success);
            }
            else {
            MessageUtilities.UpdateMessage(uxMessage,  "Content successfully published
and  its content id is : " + contentID + "and content Status is :" + cData.Status,
Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }

    }
```

Update

```
Update(Ektron.Cms.ContentData)
```

Updates an existing content item in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID
- Title
- Keywords
- Summary
- HTML

Parameters

- `contentData`. The ContentData object to update.

Returns

Returns the custom ContentData object update.

Remarks

Validate the Content item with `GetItem()` before you update the properties. Then, call `Update` with your modified ContentData object.

NOTE: You cannot change all properties after the initial Add event, such as the ContentData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxContentTitleLabel" AssociatedControlID="uxContentTitle"
CssClass="span-3 last" runat="server" Text="Title:" />
```

```
        <ektronUI:TextField  ID="uxContentTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <asp:Label ID="Label1" AssociatedControlID="uxContentKeywords" CssClass="span-3
last" runat="server" Text="Keywords:" />
        <ektronUI:TextField  ID="uxContentKeywords" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxContentSummaryLabel" AssociatedControlID="uxContentSummary"
CssClass="span-3 last" runat="server" Text="Summary:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxContentSummary"
CssClass="span-4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxContentHtmlLabel" AssociatedControlID="uxContentHtml"
CssClass="span-3 last" runat="server" Text="Html:" />
        <asp:TextBox TextMode="MultiLine" Rows="4"  ID="uxContentHtml" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
using Ektron.Cms;
using Microsoft.VisualBasic;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long contentID = long.Parse(uxContentId.Text);
            ContentManager contentManager = new ContentManager();
            ContentData cData = contentManager.GetItem(contentID);

            if (cData != null && cData.Id > 0)
            {
                if (!string.IsNullOrEmpty(uxContentKeywords.Text))
                {
                    cData.MetaData = new ContentMetaData[1];
                    cData.MetaData[0] = new ContentMetaData()
                    {
                        Id = 116,
                        Text = uxContentKeywords.Text
                    };
                }

                cData.Title = uxContentTitle.Text != string.Empty ? uxContentTitle.Text
: cData.Title;
                cData.Teaser = uxContentSummary.Text != string.Empty ?
uxContentSummary.Text : cData.Teaser;
                cData.Html = uxContentHtml.Text != string.Empty ? uxContentHtml.Text :
cData.Html;
                cData.UserId = contentManager.UserId;
                contentManager.Update(cData);
                MessageUtilities.UpdateMessage(uxMessage, "Content with ID : " +
cData.Id + " has been updated.", Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Invalid content ID ",
Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

ContentCollectionCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Content
```

Constructors

- `ContentCollectionCriteria()`

  ```
  public ContentCollectionCriteria()
  ```

- `ContentCollectionCriteria(Ektron.Cms.Common.ContentProperty, EkEnumeration.OrderByDirection)`

  ```
  public ContentCollectionCriteria(Ektron.Cms.Common.ContentProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `ContentProperty` are:

  - ContentPath
  - DateCreated
  - DateModified
  - EndDate
  - EndDateActionType
  - ExpireDate
  - ExternalTypeId
  - FolderId
  - FolderName
  - GoLivedate
  - Id
  - InheritedFromId
  - IsArchived
  - IsPublished
  - IsSearchable
  - LanguageId
  - LastEditorFirstName
  - LastEditorLastName
  - Path
  - Status
  - SubType
  - Title
  - Type
  - UserId
  - XmlConfigurationId

Methods

- `AddFilter(Ektron.Cms.Common.ContentCollectionProperty, Ektron.Cms.Common.CriteriaFilterOperator, object)`. Adds a Filter to the criteria for filtering based upon a content collection.

```
public Ektron.Cms.Content.ContentCollectionFilterGroup AddFilter
(Ektron.Cms.Common.ContentCollectionProperty field,
Ektron.Cms.Common.CriteriaFilterOperator operator, object value)
```

- `AddFilter(string)`. Adds a Filter to the criteria for filtering based upon a content collection.

```
public Ektron.Cms.Content.ContentCollectionFilterGroup AddFilter(string
collectionName)
```

- `AddFilter(long)`. Adds a Filter to the criteria for filtering based upon a content collection.

```
public Ektron.Cms.Content.ContentCollectionFilterGroup AddFilter(long collectionId)
```

- `CollectionGroupFilters`

```
public System.Collections.Generic.List<ContentCollectionFilterGroup>
CollectionGroupFilters { set; get; }
```

- `GenerateSql(System.Data.Common.DbCommand, System.Collections.Generic.Dictionary<Ektron.Cms.Common.ContentProperty,string>, out string, out string)`

```
public override void
  GenerateSql(System.Data.Common.DbCommand command,
  System.Collections.Generic.Dictionary<ContentProperty,string>
  columnMap, out string whereClause, out string orderByClause)
```

- `OrderByCollectionOrder`. If true, will disregard OrderBy property and instead use predetermined content collection order.

```
public bool OrderByCollectionOrder { set; get; }
```

- `ToCacheKey()`

```
public override string ToCacheKey()
```

ContentCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Content
```

Constructors

- `ContentCriteria()`

```
public ContentCriteria()
```

- `ContentCriteria(Ektron.Cms.Common.ContentProperty, EkEnumeration.OrderByDirection)`

```
public ContentCriteria(Ektron.Cms.Common.ContentProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `ContentProperty` are:

- ContentPath
- DateCreated
- DateModified
- EndDate
- EndDateActionType
- ExpireDate
- ExternalTypeId
- FolderId
- FolderName
- GoLivedate
- Id
- InheritedFromId
- IsArchived
- IsPublished
- IsSearchable
- LanguageId
- LastEditorFirstName
- LastEditorLastName
- Path
- Status
- SubType
- Title
- Type
- UserId
- XmlConfigurationId

ContentData

Namespace

```
Ektron.Cms
```

Properties

- `ApprovalMethod`. Gets or sets the approval method of a content item for the ContentData object. Returns an integer value representing the type of approval method. If the content is inheriting the approval method from the folder, the IsInherited property must be false to change this property. Choices are:
  - 1 = Force All Approvers. Content checked-out during the approval process will force the approval chain to reset to the beginning.
  - 0 = Do Not Force All Approvers. Content checked-out during the approval process will continue through the approval chain.

```
public int ApprovalMethod { set; get; }
```

- `Approver`. Gets or sets the approvers of a content item for the `ContentData` object. Returns a string value representing the approvers of a content item.

```
public string Approver { set; get; }
```

- `Comment`. Gets or sets comment information associated with a content item for the `ContentData` object. Returns comment information associated with a content item.

```
public string Comment { set; get; }
```

- `ContentPath`. Gets or sets the Ektron CMS path to content item for the`ContentData` object, which includes the content title. For example, `folder1/Content Title`. Returns a string value representing the Ektron CMS path to the content item, including the content's title.

```
public string ContentPath { set; get; }
```

- `ContType`. Gets or sets the content type as an integer, should be 1 for content Returns an integer representing the content type.

```
public int ContType { set; get; }
```

- `DateCreated`. Gets or sets the creation date of a content item for the `ContentData` object. Returns a creation date of a content item as DateTime.

```
public System.DateTime DateCreated { set; get; }
```

- `DateModified`. Gets or sets the content's last edit date for the `ContentData` object. Returns a content's last edit date as a DateTime.

```
public System.DateTime DateModified { set; get; }
```

- `DisplayDateCreated`. Gets or sets the creation date of a content item formatted for display for the `ContentData` object. For example, June 11, 2014. Returns a creation date of a content item formatted for display.

```
public string DisplayDateCreated { set; get; }
```

- `DisplayEndDate`. Gets or sets the end date of a content item formatted for display. This is for the `ContentData` object. For example, June 11, 2014. Returns an end date of a content item formatted for display.

```
public string DisplayEndDate { set; get; }
```

- `DisplayGoLive`. Gets or sets the start date a content item goes live on the site formatted for display. This is for the `ContentData` object. For example, June 11, 2014. Returns a start date a content item goes live on the site formatted for display.

```
public string DisplayGoLive { set; get; }
```

- `DisplayLastEditDate`. Gets or sets the content's last edit date formatted for display for the `ContentData` object. For example, June 11, 2014. Returns a content's last edit date formatted for display.

```
public string DisplayLastEditDate { set; get; }
```

- `EditorFirstName`. Gets the content editor's first name for the `ContentData` object. This property is populated on retrieval, but is not persisted when saving `ContentData`. Returns an editor's first name as a string.

```
public string EditorFirstName { set; get; }
```

- `EditorLastName`. Gets the content editor's last name for the `ContentData` object. This property is populated on retrieval, but is not persisted when saving `ContentData`. Returns an editor's last name as a string.

```
public string EditorLastName { set; get; }
```

- `EditorUserNames`. Gets the username of the last editor of the content item for the `ContentData` object. This property is populated on retrieval, but is not persisted when saving `ContentData`. Returns a username of the last editor of the content item.

```
public string EditorUserNames { set; get; }
```

- `EndDate`. Gets or sets the end date of a content item. This is for the `ContentData` object. Returns a string value that represents the end date of a content item.

```
public string EndDate { set; get; }
```

- `EndDateAction`. Gets or sets the action to perform on a content item once the End Date has been reached. This is for the `ContentData` object. Returns an integer value representing one of the above end date action choices. Choices are:
  - 1 = Archive and remove from site (expire)
  - 2 = Archive and remain on site
  - 3 = Add to the CMS Refresh Report.

```
public int EndDateAction { set; get; }
```

- `EndDateActionType`. Gets or sets the End Action type.

```
public EkEnumeration.CMSEndDateAction EndDateActionType { set; get; }
```

- `ExpireDate`. Gets or sets the expire date of a content item. This is for the `ContentData` object. Returns a string value that represents the expire date of a content item.

```
public System.DateTime ExpireDate { set; get; }
```

- `FlagDefId`. Gets or sets the content item's Flag Defition Id for the `ContentData` object. Returns a content item's Flag Definition Id as long for the `ContentData` object.

```
public long FlagDefId { set; get; }
```

- `FolderId`. Gets or sets the content item's folder ID for the `ContentData` object. Reutns a folder ID as a long.

```
public long FolderId { set; get; }
```

- `FolderName`. Gets or sets the content item's folder name for the `ContentData` object. Returns the content item's folder name.

```
public string FolderName { set; get; }
```

- `GoLive`. Gets or sets the start date a content item goes live on the site. This is for the `ContentData` object. Returns a string value that represents the date a content item goes live.

```
public string GoLive { set; get; }
```

- `GoLiveDate`. Gets or sets the Golive date a content item goes live on the site. This is for the `ContentData` object. Returns a string value that represents the date a content item goes live.

```
public System.DateTime? GoLiveDate { set; get; }
```

- `HistoryId`. Gets or sets the content item's history ID for the `ContentData` object. Returns the numeric ID (Long) of the content item's history.

```
public long HistoryId { set; get; }
```

- `HyperLink`. Gets or sets the content item's Hyperlink in the Ektron CMS for the`ContentData` object. Returns a content item's Hyperlink information.

```
public string HyperLink { set; get; }
```

- `InheritedFrom`. Gets or sets the folder ID from which to inherit permissions for the `ContentData`object. Returns a numeric ID representing the folder from which permissions are inherited.

```
public long InheritedFrom { set; get; }
```

- `IsInherited`. Gets or sets whether the content item inherits permissions from the folder settings for the `ContentData` object. Returns a boolean that returns true when the content item inherits permissions from the folder settings.

```
public bool IsInherited { set; get; }
```

- `IsMetaComplete`. This property is deprecated.

```
public bool IsMetaComplete { set; get; }
```

- `IsPermissionsInherited`. Gets or sets whether the content item inherits permissions from the folder settings for the `ContentData` object. Returns a boolean that returns true when the content item inherits permissions from the folder settings.

```
public bool IsPermissionsInherited { set; get; }
```

- `IsPublished`. Gets or sets whether the content is published before for the `ContentData` object. Returns a boolean based on whether the content is published before. True = Published

```
public bool IsPublished { set; get; }
```

- `IsSearchable`. Gets or sets whether the content is searchable for the `ContentData` object. Returns a boolean based on whether the content is searchable.
  - True = content is searchable.
  - False = content is not searchable.

```
public bool? IsSearchable { set; get; }
```

- `IsSearchableAsBool`. XML comment contains invalid XML: End tag 'returns' does not match the start tag 'summary'.

```
public bool IsSearchableAsBool { set; get; }
```

- `IsXmlInherited`. Gets or sets whether the XML Smart Form configuration is inherited from the folder. This is for the `ContentData` object. Returns a boolean based on whether the XML Smart Form configuration is inherited from the folder.

- True = content XML Smart Form configuration is inherited from the folder.
- False = content XML Smart Form configuration is not inherited from the folder.

```
public bool IsXmlInherited { set; get; }
```

- `LanguageDescription`

```
public string LanguageDescription { set; get; }
```

- `LastEditDate`. This property is deprecated: Use DateModified, which returns a DateTime rather than a string. Gets or sets the content's last edit date for the `ContentData` object. Returns a content's last edit date as a string.

```
public string LastEditDate { set; get; }
```

- `LegacyData`. Gets or sets the Legacy Data of a content item as an Object of `ContentMetadata`. This is for the `ContentData` object. Returns Legacy Data as an Object of `ContentMetaData`.

```
public object LegacyData { set; get; }
```

- `ManualAlias`. Gets or sets the manual alias by its name for the `ContentData` object. For example, `/~YourSite~/text.aspx`. Returns a string value representing the name of a manual alias.

```
public string ManualAlias { set; get; }
```

- `ManualAliasId`. Gets or sets the ID of the of a manual alias for the `ContentData` object. Returns a numeric value (`Long`) representing a manual alias ID.

```
public long ManualAliasId { set; get; }
```

- `MediaText`. Gets or sets any `MediaText` associated with the `ContentData` object. Returns Media Text associated with `ContentData` object item.

```
public string MediaText { set; get; }
```

- `MetaData`. Gets or sets the Metadata of a content item as an array of `ContentMetadata`. This is for the `ContentData` object. Returns metadata as an array of `ContentMetaData`.

```
public Ektron.Cms.ContentMetaData[] MetaData { set; get; }
```

- `Path`. Gets or sets the Ektron CMS path to content item for the`ContentData`object. Returns a string value representing the Ektron CMS path to the content item.

```
public string Path { set; get; }
```

- `PermissionInheritedFrom`. Gets or sets the folder ID from which to inherit permissions for the `ContentData` object. Returns a numeric ID representing the folder from which permissions are inherited.

```
public long PermissionInheritedFrom { set; get; }
```

- `Permissions`. Gets or sets permission data for the `ContentData` object. Permissions as a `PermissionData` object.

```
public Ektron.Cms.PermissionData Permissions { set; get; }
```

- `Status`. This property should be considered read-only. This value is managed by the Ektron approval workflow (if any) attached to the item. Gets the letter value

representing the status of a content item for the `ContentData` object. Returns a letter value representing the status of a content item.

- A = Approved
- D = Pending Deletion.
- I = Checked In
- M = Marked for Deletion
- O = Checked Out
- P = Pending Go Live Date
- S = Submitted for Approval
- T = Awaiting Completion of Associated Tasks

```
public string Status { set; get; }
```

- `StyleSheet`. Gets or sets the style sheet to use with a content item for the `ContentData` object. Returns a string value representing the style sheet to use with a content item.

```
public string StyleSheet { set; get; }
```

- `TemplateCnfiguration`. Gets or sets the content item's template configuration for the `ContentData` object. Returns a content item's template configuration as `TemplateData`.

```
public Ektron.Cms.TemplateData TemplateConfiguration { set; get; }
```

- `Text`. Gets or sets a string value representing the Ektron CMS`ContentData`object. Returns a string value representing the Ektron CMS`ContentData` object.

```
public string Text { set; get; }
```

- `Updates`. Gets or sets the updates information associated with the `ContentData` object. Returns updates of type `Integer`.

```
public int Updates { set; get; }
```

- `UserId`. Gets or sets the ID of the user for the `ContentData` object. Returns a User ID as a `Long`.

```
public long UserId { set; get; }
```

- `UserName`. This property is obsolete. To retrieve the user name, call `EkUser GetActiveUserbyID` with the correct User ID. Returns the content's username. Gets or sets the content's username for the `ContentData` object.
-

```
public string UserName { set; get; }
```

- `Validate()`. Validation method for this object; returns a `ValidationResults` instance.

```
public override ValidationResults Validate()
```

- `XmlConfiguration`. Gets or sets the content item's XML configuration for the `ContentData` object. Returns a content item's XML configuration as `XmlConfigData`.

```
public XmlConfigData XmlConfiguration { set; get; }
```

- `XmlInheritedFrom`. Gets or sets the ID of the folder from which the XML Smart Form configuration is inherited. This is for the `ContentData` object. Returns a

numeric ID (`Long`) of the folder from which the XML Smart Form configuration is inherited.

```
public long XmlInheritedFrom { set; get; }
```

ContentMetadataCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Content
```

Constructors

- `ContentMetadataCriteria()`

  ```
  public ContentMetadataCriteria()
  ```

- `ContentMetadataCriteria(Ektron.Cms.Common.ContentProperty, EkEnumeration.OrderByDirection)`

  ```
  public ContentMetadataCriteria(Ektron.Cms.Common.ContentProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `ContentProperty` are:

  - ContentPath
  - DateCreated
  - DateModified
  - EndDate
  - EndDateActionType
  - ExpireDate
  - ExternalTypeId
  - FolderId
  - FolderName
  - GoLivedate
  - Id
  - InheritedFromId
  - IsArchived
  - IsPublished
  - IsSearchable
  - LanguageId
  - LastEditorFirstName
  - LastEditorLastName
  - Path
  - Status
  - SubType
  - Title

- Type
- UserId
- XmlConfigurationId

Methods

- `AddFilter(ContentMetadatatProperty, operator, value)`. Filters the results based on the condition defined in this method.

```
criteria.addFilter(ContentMetadataProperty field, CriteriaFilterOperator
filterOperator , object value );
```

- `GenerateSql(System.Data.Common.DbCommand, System.Collections.Generic.Dictionary<Ektron.Cms.Common.ContentProperty,string>, out string, out string)`

```
public override void
  GenerateSql(System.Data.Common.DbCommand command,
  System.Collections.Generic.Dictionary<ContentProperty,string>
  columnMap, out string whereClause, out string orderByClause)
```

- `ToCacheKey()`

```
public override string ToCacheKey()
```

- `MetadataFilterGroups`

```
public System.Collections.Generic.List<ContentMetadataFilterGroup>
  MetadataFilterGroups { set; get; }
```

ContentTaxonomyCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Content
```

Constructors

- `ContentTaxonomyCriteria()`

```
public ContentTaxonomyCriteria()
```

- `ContentTaxonomyCriteria(Ektron.Cms.Common.ContentProperty, EkEnumeration.OrderByDirection)`

```
public ContentTaxonomyCriteria(Ektron.Cms.Common.ContentProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `ContentProperty` are:

- ContentPath
- DateCreated
- DateModified
- EndDate
- EndDateActionType
- ExpireDate

- ExternalTypeId
- FolderId
- FolderName
- GoLivedate
- Id
- InheritedFromId
- IsArchived
- IsPublished
- IsSearchable
- LanguageId
- LastEditorFirstName
- LastEditorLastName
- Path
- Status
- SubType
- Title
- Type
- UserId
- XmlConfigurationId

Methods

- `AddFilter(string, [bool])`

```
public Ektron.Cms.Content.ContentTaxonomyFilterGroup
 AddFilter(string taxonomyPath, [bool recursive = false])
```

- `AddFilter(long, [bool])`

```
public Ektron.Cms.Content.ContentTaxonomyFilterGroup
 AddFilter(long taxonomyId, [bool recursive = false])
```

- `GenerateSql(System.Data.Common.DbCommand, System.Collections.Generic.Dictionary<Ektron.Cms.Common.ContentProperty,string>, out string, out string)`

```
public override void
 GenerateSql(System.Data.Common.DbCommand command,
 System.Collections.Generic.Dictionary<ContentProperty,string>
 columnMap, out string whereClause, out string orderByClause)
```

- `TaxonomyGroupFilters`

```
public System.Collections.Generic.List<ContentTaxonomyFilterGroup>
 TaxonomyGroupFilters { set; get; }
```

- `ToCacheKey()`

```
public override string ToCacheKey()
```

ContentRatingManager

8.50 and higher

The ContentRatingManager class manages user valuations on content.

Namespace

```
Ektron.Cms.Framework.Content.ContentRatingManager
```

Constructors

- `ContentRatingManager()`
- `ContentRatingManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `ContentRatingService`. Returns an instance of the business objects Content Rating Service.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.contentRatingTypeData)
```

Adds a content rating type.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID
- * User Rating
- User Comments

Parameters

- `contentRating`. The ContentRatingData object.

Returns

The custom ContentRatingData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIDLabel" AssociatedControlID="uxContentID"
CssClass="span-3 last" runat="server" Text="Content Id:" />
        <ektronUI:TextField ID="uxContentID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
        </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxUserRatingLabel" AssociatedControlID="uxUserRating"
CssClass="span-3 last" runat="server" Text="User Rating(between 1 and 10 and no decimal
values):" />
        <ektronUI:TextField ID="uxUserRating" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
        </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxUserCommentsLabel" AssociatedControlID="uxUserComments"
CssClass="span-3 last" runat="server" Text="User Comments:" />
        <ektronUI:TextField ID="uxUserComments" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
        </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxContentRatingStatus" Visible="false" CssClass="span-3
last" runat="server" Text="Status:" />
        </li>
        <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
        </li>
```

```
</ol>
<ol class="formFields">
        <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
        </li>
        <li class="clearfix">
        <asp:Literal ID="uxContentLanguage" runat="server"></asp:Literal>
        </li>
        <li class="clearfix">
        <asp:Literal ID="uxUserID" runat="server"></asp:Literal>
        </li>
        <li class="clearfix">
        <asp:Literal ID="uxLilUserRating" runat="server"></asp:Literal>
        </li>
        <li class="clearfix">
        <asp:Literal ID="uxRatingDate" runat="server"></asp:Literal>
        </li>
        <li class="clearfix">
        <asp:Literal ID="uxUserComment" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.User;
using Ektron.Cms;
using Microsoft.VisualBasic;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
            try
            {
                ContentRatingManager contentRatingManager = new ContentRatingManager();
                ContentRatingData cRatingData = new ContentRatingData();
                long contentID = long.Parse(uxContentID.Text);
                bool cIDResult = Information.IsNumeric(uxContentID.Text);
                int contentRating = int.Parse(uxUserRating.Text);
                bool cRatingResult = Information.IsNumeric(uxContentID.Text);
                if (!(cRatingResult))
                {
                    MessageUtilities.UpdateMessage(uxMessage, " Rating Should be between
0 and 10", Message.DisplayModes.Error);
                    return;
                }
```

```
                string userComments = uxUserComments.Text;
                UserAPI userApi = new UserAPI();
                long UserId = userApi.RequestInformationRef.UserId;
                ContentManager contentManager = new ContentManager();
                ContentData cData = new ContentData();
                cData = contentManager.GetItem(contentID, false);
                if (!(cIDResult) && (cData == null))
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Content Id is not valid",
Message.DisplayModes.Error);
                    return;
                }
                cRatingData.ContentId = contentID;
                cRatingData.ContentLanguageId = cData.LanguageId;
                cRatingData.UserComments = userComments;
                cRatingData.UserId = UserId;
                cRatingData.UserRating = contentRating;
                cRatingData.RatingDate = DateTime.Now;
                cRatingData.VisitorId = userApi.RequestInformationRef.ClientEktGUID;
                contentRatingManager.Add(cRatingData);

                MessageUtilities.UpdateMessage(uxMessage, "Content ID : " +
cRatingData.ContentId, Message.DisplayModes.Success);

                uxContentLanguage.Text = "ContentLanguage Id : " +
cRatingData.ContentLanguageId;
                if (UserId > 0) { uxUserID.Text = "Content Reviewed UserId : " +
cRatingData.UserId; }
                uxLilUserRating.Text = "User Rating : " + cRatingData.UserRating;
                uxRatingDate.Text = "RatingDate : " + cRatingData.RatingDate;
                uxUserComment.Text = "UserComments : " + cRatingData.UserComments;
                uxPageMultiView.SetActiveView(uxViewMessage);
            }
            catch (Exception ex)
            {
                MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
                uxPageMultiView.SetActiveView(uxViewMessage);
            }

        }
```

Delete

```
Delete(System.Int64)
```

Deletes a content rating type from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content Rating ID

Parameters

- `contentRatingId`. The ID of the content rating type to be deleted.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentRatingIdLabel"
AssociatedControlID="uxContentRatingId" CssClass="span-3 last" runat="server" Text="*
ContentRating Id:" />
        <ektronUI:TextField ID="uxContentRatingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentRatingStatus" Visible="false" CssClass="span-3
last" runat="server" Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
using Microsoft.VisualBasic;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
   {
    try
     {
        ContentRatingManager contentRatingManager = new ContentRatingManager();
        ContentRatingData cRatingData = new ContentRatingData();
        long contentRatingID = long.Parse(uxContentRatingId.Text);
        bool result = Information.IsNumeric(uxContentRatingId.Text);
        cRatingData = contentRatingManager.GetItem(contentRatingID);
        if (!((result) && cRatingData != null))
```

```
        {
            uxStatus.Visible = true;
            MessageUtilities.UpdateMessage(uxStatus,"Invalid content Rating ID ",
Message.DisplayModes.Error);
            return;
        }

        contentRatingManager.Delete(contentRatingID);

        MessageUtilities.UpdateMessage(uxMessage, "The following content rating ID
successfully deleted  : " + contentRatingID, Message.DisplayModes.Success);

        uxPageMultiView.SetActiveView(uxViewMessage);
     }
     catch (Exception ex)
     {
         MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
         uxPageMultiView.SetActiveView(uxViewMessage);
     }

   }
```

GetItem

```
GetItem(Ektron.Cms.ContentData)
```

Retrieves the properties of a specific content rating object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content Rating ID

Parameters

- `contentRatingId`. The ID of the content rating type to get.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentRatingIdLabel"
AssociatedControlID="uxContentRatingId" CssClass="span-3 last" runat="server" Text="*
ContentRating Id:" />
        <ektronUI:TextField ID="uxContentRatingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxContentRatingStatus" Visible="false" CssClass="span-3
last" runat="server" Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetDetails"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
 </ol>
 <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentLanguage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentReviewState" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserRating" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxRatingDate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserComment" runat="server"></asp:Literal>
    </li>
 </ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
using Microsoft.VisualBasic;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

            try
            {
```

```
                ContentRatingManager contentRatingManager = new ContentRatingManager();
                ContentRatingData cRatingData = new ContentRatingData();
                long contentRatingID = long.Parse(uxContentRatingId.Text);
                bool result = Information.IsNumeric(uxContentRatingId.Text);
                cRatingData = contentRatingManager.GetItem(contentRatingID);
                if (!((result) && cRatingData != null))
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Invalid content Rating ID
", Message.DisplayModes.Error);
                    return;
                }

                MessageUtilities.UpdateMessage(uxMessage, "Content rating details for
content rating Id : " + contentRatingID + " are below", Message.DisplayModes.Success);
                uxContentID.Text = "Content ID : " + cRatingData.ContentId ;
                uxContentLanguage.Text = "ContentLanguage Id : " +
cRatingData.ContentLanguageId ;
                uxContentReviewState.Text= "ContentReviewState : " +
cRatingData.ContentReviewState ;
                uxUserID.Text = "Content Reviewed UserId : " + cRatingData.UserId ;
                uxUserRating.Text = "User Rating : " + cRatingData.UserRating ;
                uxRatingDate.Text = "RatingDate : " + cRatingData.RatingDate ;
                uxUserComment.Text = "UserComments : " + cRatingData.UserComments;
                uxPageMultiView.SetActiveView(uxViewMessage);
            }
            catch (Exception ex)
            {
                MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
                uxPageMultiView.SetActiveView(uxViewMessage);
            }


    }
```

GetList

```
GetList(Ektron.Cms.Content.ContentRatingCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content Rating Property
- * Object Value

Parameters

- `criteria`. ContentRatingCriteria used to retrieve content.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentRatingPropertyLabel"
AssociatedControlID="uxContentRatingProperty" CssClass="span-6 last" runat="server"
Text="* ContentRatingProperty:" />
        <asp:DropDownList ID="uxContentRatingProperty" runat="server">
            <asp:ListItem>UserId</asp:ListItem>
            <asp:ListItem>ContentLanguageId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxContentRatingDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Content Id
                    </th>
                        <th>
                        User Rating
                    </th>
                    <th>
                        User Comments
                    </th>
                    <th>
                        Rating Date
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
```

```
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("ContentId")%>
            </td>
                <td class="devsite-method">
                <%# Eval("UserRating")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserComments")%>
            </td>
            <td class="devsite-method">
                <%# Eval("RatingDate")%>
            </td>

        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Content;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            ContentRatingCriteria contentRatingCriteria = new ContentRatingCriteria
(ContentRatingProperty.Id, EkEnumeration.OrderByDirection.Descending);
            if (uxContentRatingProperty.SelectedItem.Text == "UserId")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                contentRatingCriteria.AddFilter(ContentRatingProperty.UserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = int.Parse(uxObjectValue.Text);
                contentRatingCriteria.AddFilter(ContentRatingProperty.ContentLanguageId,
```

```
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            ContentRatingManager contentRatingManager = new ContentRatingManager();
            List<ContentRatingData> contentRatingDataList = contentRatingManager.GetList
(contentRatingCriteria);

            uxContentRatingDataListView.Visible = true;
            uxContentRatingDataListView.DataSource = contentRatingDataList;
            uxContentRatingDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Purge

```
Purge(System.Int64,System.DateTime,System.DateTime)
```

Purges all ratings of a particular content block over a defined date/time range.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID
- * Start Date Time
- * End Date Time

Parameters

- `contentId`. The ID of the content that is having its ratings purged.
- `startDate`. The beginning of the date range of content to be purged.
- `endDate`. The end of the date range of content to be purged.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
```

```
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-3 last" runat="server" Text="*Start DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-6" runat="server"
Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
        </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxEndDateLabel" AssociatedControlID="uxEndDate"
CssClass="span-3 last" runat="server" Text="*End DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxEndDate" CssClass="span-6" runat="server"
Text="MM/DD/YYYY" ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxEndTime" runat="server">
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
            <asp:ListItem>12</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEndPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Purge"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            String st = uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00
" + uxStartPeriod.SelectedItem.Text;
            String et = uxEndDate.Text + " " + uxEndTime.SelectedItem.Text + ":00:00 " +
uxEndPeriod.SelectedItem.Text;
            DateTime startDateTime = DateTime.Parse(st);
            DateTime endDateTime = DateTime.Parse(et);

            ContentRatingManager contentRatingManager = new ContentRatingManager();
            contentRatingManager.Purge(long.Parse(uxContentId.Text) , startDateTime,
endDateTime);

            MessageUtilities.UpdateMessage(uxMessage, "The content rating successfully
purged.", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Update

```
Update(Ektron.Cms.ContentRatingData)
```

Updates an existing content rating item in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content Rating ID
- * User Rating
- User Comments

Parameters

- `contentRating`. The ContentRatingtData object.

Returns

The custom ContentRatingData object updated.

Remarks

Validate the content rating item with `GetItem()` before you update the properties. Then, call `Update` with your modified ContentRatingData object.

NOTE: You cannot change all properties after the initial Add event, such as the ContentRatingData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentRatingIdLabel"
AssociatedControlID="uxContentRatingId" CssClass="span-3 last" runat="server" Text="*
ContentRating Id:" />
        <ektronUI:TextField ID="uxContentRatingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="1" />
    </li>
    <li class="clearfix">
        <asp:Label ID="Label1" Visible="false" CssClass="span-3 last" runat="server"
Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserRatingLabel" AssociatedControlID="uxUserRating"
CssClass="span-3 last" runat="server" Text="* User Rating(between 1 and 10 and no
decimal values):" />
        <ektronUI:TextField ID="uxUserRating" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="9" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserCommentsLabel" AssociatedControlID="uxUserComments"
CssClass="span-3 last" runat="server" Text="User Comments:" />
        <ektronUI:TextField ID="uxUserComments" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentRatingStatus" Visible="false" CssClass="span-3
last" runat="server" Text="Status:" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button ID="Button1" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="Label2" CssClass="span-4" runat="server" Text="* - Required"
/>
    </li>
 </ol>
 <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentLanguage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLilUserRating" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxRatingDate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserComment" runat="server"></asp:Literal>
    </li>
 </ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.User;
using Ektron.Cms;
using Microsoft.VisualBasic;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ContentRatingManager contentRatingManager = new ContentRatingManager();
            ContentRatingData cRatingData = new ContentRatingData();
            long contentRatingID = long.Parse(uxContentRatingId.Text);
            bool result = Information.IsNumeric(uxContentRatingId.Text);
```

```
            cRatingData = contentRatingManager.GetItem(contentRatingID);
            if (!((result) && cRatingData != null))
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Invalid content Rating ID ",
Message.DisplayModes.Error);
                return;
            }
            int contentRating = int.Parse(uxUserRating.Text);
            bool cRatingResult = Information.IsNumeric(contentRating);
            if (!(cRatingResult))
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, " Rating Should be between 0
and 10", Message.DisplayModes.Error);
                return;
            }
            string userComments = uxUserComments.Text;
            UserAPI userApi = new UserAPI();
            long UserId = userApi.RequestInformationRef.UserId;
            cRatingData.Id = contentRatingID;
            cRatingData.UserComments = userComments;
            cRatingData.UserId = UserId;
            cRatingData.RatingDate = DateTime.Now;
            cRatingData.UserRating = contentRating;
            cRatingData.VisitorId = userApi.RequestInformationRef.ClientEktGUID;
            contentRatingManager.Update(cRatingData);

            MessageUtilities.UpdateMessage(uxMessage, "Content Rating ID : " +
contentRatingID, Message.DisplayModes.Success);
            uxContentID.Text = "Content ID : "  + cRatingData.ContentId;
            uxContentLanguage.Text = "ContentLanguage Id : " +
cRatingData.ContentLanguageId;
            if (UserId > 0) { uxUserID.Text = "Content Reviewed UserId : " +
cRatingData.UserId; }
            uxLilUserRating.Text = "User Rating : " + cRatingData.UserRating;
            uxRatingDate.Text = "RatingDate : " + cRatingData.RatingDate;
            uxUserComment.Text = "UserComments : " + cRatingData.UserComments;
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

ContentRatingCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Content
```

Properties

- `ContentRatingCriteria()`

  ```
  public ContentRatingCriteria()
  ```

- `ContentRatingCriteria(Ektron.Cms.Common.ContentRatingProperty, EkEnumeration.OrderByDirection)`

  ```
  public ContentRatingCriteria(Ektron.Cms.Common.ContentRatingProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `ContentRatingProperty` are:

  - ContentId
  - ContentLanguageId
  - ContentReviewState
  - Id
  - RatingDate
  - RatingState
  - UserId
  - VistorId

ContentRatingData

Namespace

```
Ektron.Cms
```

Properties

- `ContentId`

  ```
  public long ContentId { set; get; }
  ```

- `ContentLanguageId`

  ```
  public int ContentLanguageId { set; get; }
  ```

- `ContentReviewState`

  ```
  public Ektron.Cms.Common.EkEnumeration.ContentReviewState
    ContentReviewState { set; get; }
  ```

- `RatingDate`

  ```
  public System.DateTime RatingDate { set; get; }
  ```

- `UserComments`

  ```
  public string UserComments { set; get; }
  ```

- `UserId`

  ```
  public long UserId { set; get; }
  ```

- `UserRating`

  ```
  public int UserRating { set; get; }
  ```

- VisitorId

```
public string VisitorId { set; get; }
```

LibraryManager

8.50 and higher

The LibraryManager class manages libraries, which stores images, files, quicklinks, and hyperlinks that can be inserted into editor content.

Namespace

```
Ektron.Cms.Framework.Content.LibraryManager
```

Constructors

- `LibraryManager()`
- `LibraryManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `LibraryManagerService`. Returns an instance of the business objects library manager service.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 1146
- *GetItem* on page 1147
- `GetLibraryItemByContentId(Int64,Int32)`. 8.60 and higher Retrieves a `LibraryData` object based upon the content ID.
- *GetList* on page 1149
- *Update* on page 1152

Add

```
Add(Ektron.Cms.LibraryData)
```

Adds a library based on information in a LibraryData object.

The method populates the message.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID
- * Type ID
- * Title
- * File Name
- * Language ID

Parameters

- `libraryData`. The LibraryData object to add.

Returns

Returns the custom LibraryData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last"
            runat="server" Text="*Folder Id :" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <div class="ektronTopSpace">
    </div>
    <li class="clearfix">
        <ektronUI:Label ID="uxTypeIdLabel" AssociatedControlID="uxTypeId"
CssClass="span-3 last"
            runat="server" Text="*TypeId :" />
        <asp:DropDownList ID="uxTypeId" runat="server">
            <asp:ListItem Value="1">images</asp:ListItem>
            <asp:ListItem Value="2">files</asp:ListItem>
        </asp:DropDownList>
    </li>
    <div class="ektronTopSpace">
    </div>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxLibraryTitle"
CssClass="span-3 last"
            runat="server" Text="*Title :" />
        <ektronUI:TextField ID="uxLibraryTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace">
    </div>
    <li class="clearfix">
        <ektronUI:Label ID="uxFileNameLabel" AssociatedControlID="uxFileName"
CssClass="span-3 last"
            runat="server" Text="*File Name :" />
        <asp:FileUpload ID="uxFileName" size="29.75" CssClass="span-4" runat="server" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageIdLabel" AssociatedControlID="uxLanguageId"
CssClass="span-3 last"
            runat="server" Text="*Language Id :" />
        <ektronUI:TextField ID="uxLanguageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1033" />
    </li>
    <div class="ektronTopSpace">
    </div>
    <li class="clearfix">
        <ektronUI:Message ID="uxStatus" DisplayMode="Information" Visible="false"
runat="server" />
    </li>
    <div class="ektronTopSpace">
    </div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            MemoryStream stream = null;
            byte[] byteArray = null;
            HttpPostedFile postedFile = uxFileName.PostedFile;
            int fileLength = postedFile.ContentLength;
            byte[] fileData = new byte[fileLength];
            postedFile.InputStream.Read(fileData, 0, fileLength);
            if (fileData.Length > 0)
            {
                stream = new MemoryStream(fileData);
                byteArray = new byte[stream.Length];
                stream.Position = 0;
                stream.Read(byteArray, 0, (int)stream.Length);

            }
            else
            {
                MessageUtilities.UpdateMessage(uxStatus, "Please Upload a file.(A
filename is required for this library item.)", Message.DisplayModes.Error);
                return;
            }

            long folderId = long.Parse(uxFolderId.Text);
            Ektron.Cms.API.Library libraryAPI = new Ektron.Cms.API.Library();


            LibraryManager libraryManager = new LibraryManager();
            LibraryConfigData lib_setting_data = libraryAPI.GetLibrarySettings
(folderId); ;

            string strFilename = "";

            if (uxTypeId.SelectedItem.ToString() == "images")
            {

                strFilename = Server.MapPath(lib_setting_data.ImageDirectory) +
Path.GetFileName(postedFile.FileName);
            }
            else
            {
                strFilename = Server.MapPath(lib_setting_data.FileDirectory) +
Path.GetFileName(postedFile.FileName);
            }

            LibraryData libraryData = new LibraryData()
            {
                Title = uxLibraryTitle.Text,
                ParentId = folderId,
                FileName = strFilename,
                LanguageId = Convert.ToInt32(uxLanguageId.Text),
                TypeId = Convert.ToInt32(uxTypeId.SelectedValue),
                File = byteArray
            };

            libraryManager.Add(libraryData);
```

```
            MessageUtilities.UpdateMessage(uxMessage, "Library Item Added with Id " +
libraryData.Id, Message.DisplayModes.Success);


            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a library from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Library ID

Parameters

- `id`. The ID of the library to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxLibraryIdLabel" AssociatedControlID="uxLibraryId"
CssClass="span-6 last" runat="server" Text="Library Id:" />
        <ektronUI:TextField ID="uxLibraryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>

    <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long libraryID = long.Parse(uxLibraryId.Text);

            LibraryManager libraryeManager = new LibraryManager();
            libraryeManager.Delete(libraryID);

            MessageUtilities.UpdateMessage(uxMessage, "Library Item with Id " +
libraryID + " deleted", Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem. Retrieves a single LibraryData object from the CMS.

GetItem

```
GetItem(System.Int64)
```

Retrieves a single LibraryData object from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Library ID

Parameters

- `id`. The ID of the library to retrieve.

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxLibraryIdLabel" AssociatedControlID="uxLibraryId"
CssClass="span-6 last" runat="server" Text="Library Id :" />
            <ektronUI:TextField ID="uxLibraryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
        </li>

        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
</ol>

<ol class="formFields">
        <li class="clearfix">
            <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
        </li>
        <li class="clearfix">
            <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
        </li>
        <li class="clearfix">
            <asp:Literal ID="uxFileName" runat="server"></asp:Literal>
        </li>
        <li class="clearfix">
            <asp:Literal ID="uxLanguageId" runat="server"></asp:Literal>
        </li>
        <li class="clearfix">
            <asp:Literal ID="uxFolderId" runat="server"></asp:Literal>
        </li>
        <li class="clearfix">
            <asp:Literal ID="uxLastEditDate" runat="server"></asp:Literal>
        </li>
    </ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long LibraryID = long.Parse(uxLibraryId.Text);

            LibraryManager libraryManager = new LibraryManager();
            LibraryData libraryData = libraryManager.GetItem(LibraryID);
```

```
            if (libraryData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for Library Item with
ID " + libraryData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxTitle.Text = "Title : " + libraryData.Title;
                uxFileName.Text = "File Name : " + libraryData.FileName;
                uxLanguageId.Text = "Language ID : " + libraryData.LanguageId;
                uxFolderId.Text = "Folder ID : " + libraryData.ParentId;
                uxLastEditDate.Text = "Last Edit Date : " + libraryData.LastEditDate;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Library Item does not exist.
", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Content.LibraryCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Library Data Property
- * Object Value

Parameters

- `criteria`. LibraryCriteria used to retrieve libraries.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxLibraryDataPropertyLabel"
AssociatedControlID="uxLibraryDataProperty" CssClass="span-6 last" runat="server"
Text="* LibraryDataProperty:" />
        <asp:DropDownList ID="uxLibraryDataProperty" runat="server">
            <asp:ListItem>ContentType</asp:ListItem>
            <asp:ListItem>TypeId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxLibraryDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                        <th>
                        Title
                    </th>
                    <th>
                        File Name
                    </th>
                    <th>
                        Type
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
```

```
                <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FileName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Type")%>
            </td>

        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Content;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            LibraryManager libraryManager = new LibraryManager();
            LibraryCriteria criteria = new LibraryCriteria(LibraryProperty.Id,
EkEnumeration.OrderByDirection.Descending);
            Objectvalue = int.Parse(uxObjectValue.Text);

            if (uxLibraryDataProperty.SelectedItem.Text == "ContentType")
            {
                criteria.AddFilter(LibraryProperty.ContentType,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxLibraryDataProperty.SelectedItem.Text == "UserId")
            {
                criteria.AddFilter(LibraryProperty.UserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                criteria.AddFilter(LibraryProperty.TypeId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            List<LibraryData> libraryDataList = libraryManager.GetList(criteria);
```

```
            uxLibraryDataListView.Visible = true;
            uxLibraryDataListView.DataSource = libraryDataList;
            uxLibraryDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.LibraryData)
```

Updates an existing LibraryData item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Library ID
- Title
- File Name

Parameters

- `libraryData`. The LibraryData object to update.

Returns

Returns the custom LibraryData object updated

Remarks

Validate the library item with `GetItem()` before you update the properties. Then, call `Update` with your modified LibraryData object.

NOTE: You cannot change all properties after the initial Add event, such as the LibraryData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxLibraryIdLabel" AssociatedControlID="uxLibraryId"
```

```
CssClass="span-3 last" runat="server" Text="* Library Id :" />
        <ektronUI:TextField ID="uxLibraryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxLibraryTitle"
CssClass="span-3 last" runat="server" Text="Title :" />
        <ektronUI:TextField ID="uxLibraryTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
            <ektronUI:Label ID="uxFileNameLabel" AssociatedControlID="uxFileName"
CssClass="span-3 last" runat="server" Text="File Name :" />
            <asp:FileUpload ID="uxFileName" size="29.75" CssClass="span-4"
runat="server" />
    </li>
    <li class="clearfix">
            <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
            <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server"
Text="* - Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MemoryStream stream = null;
            HttpPostedFile postedFile = uxFileName.PostedFile;
            int fileLength = postedFile.ContentLength;
            byte[] fileData = new byte[fileLength];
            postedFile.InputStream.Read(fileData, 0, fileLength);
            if (fileData.Length > 0)
            {
                stream = new MemoryStream(fileData);
            }

            long libraryId = long.Parse(uxLibraryId.Text);
```

```
            LibraryManager libraryManager = new LibraryManager();
            LibraryData libraryData = libraryManager.GetItem(libraryId);

            if (libraryData != null)
            {
                libraryData.Title = (uxLibraryTitle.Text != string.Empty ?
uxLibraryTitle.Text : libraryData.Title);
                if (stream != null && stream.Length > 0)
                {
                    byte[] byteArray = new byte[stream.Length];
                    stream.Position = 0;
                    stream.Read(byteArray, 0, (int)stream.Length);
                    libraryData.File = byteArray;
                    libraryData.FileName = Path.GetFileName(postedFile.FileName);
                }

                libraryManager.Update(libraryData);

                MessageUtilities.UpdateMessage(uxMessage, "Library Item with Id " +
libraryData.Id + " updated.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Library Item does not exist",
Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

LibraryCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Content
```

Properties

- `LibraryCriteria()`

  ```
  public LibraryCriteria()
  ```

- `LibraryCriteria(Ektron.Cms.Common.LibraryProperty, EkEnumeration.OrderByDirection)`

```
public LibraryCriteria(Ektron.Cms.Common.LibraryProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `LibraryProperty` are:

- ContentId
- ContentType
- CreatedDate
- Id
- LanguageId
- LastEditDate
- ParentId
- TypeId
- UserId

LibraryData

Namespace

```
Ektron.Cms
```

Properties

- `ContentId`. Gets or sets the content's ID for the `LibraryData` object. Returns the content's long ID.

  ```
  public long ContentId { set; get; }
  ```

- `ContentSubType`. Gets or sets the content sub type for the LibraryData object. Returns the content sub type.

  ```
  public Ektron.Cms.Common.EkEnumeration.CMSContentSubtype
    ContentSubType { set; get; }
  ```

- ContentType. Gets or sets the content type for the LibraryData object. Possible values are defined in Ektron.Cms.Common.EkEnumeration.CMSContentType enumeration. Returns the content type.

  ```
  public int ContentType { set; get; }
  ```

- `DateCreated`. Gets or sets the date created for the LibraryData object. Returns the date created.

  ```
  public System.DateTime DateCreated { set; get; }
  ```

- `DisplayDateCreated`. Gets or sets the date created to show for the LibraryData object. Returns the date created to show.

  ```
  public string DisplayDateCreated { set; get; }
  ```

- `DisplayLastEditDate`. Gets or sets the last edit date to show for the LibraryData object. Returns the last edit date to show.

  ```
  public string DisplayLastEditDate { set; get; }
  ```

- `EditorFirstName`. Gets or sets the Editor's first name for the LibraryData object. Returns the Editor's first name.

  ```
  public string EditorFirstName { set; get; }
  ```

- `EditorLastName`. Gets or sets the Editor's last name for the LibraryData object. Returns the Editor's last name.

```
public string EditorLastName { set; get; }
```

- `FolderName`. Gets or sets the folder's name for the LibraryData object. Returns the folder's name.

```
public string FolderName { set; get; }
```

- `Html`. Gets or sets the HTML for this library item. Many library items will not have corresponding HTML.

```
public string Html { set; get; }
```

- `IsPrivate`. This property should be considered read-only. The value will be set to match the items parent folder privacy setting. Returns true if the library is private. Otherwise, returns false.

```
public bool IsPrivate { set; get; }
```

- `LastEditDate`. Gets or sets the last edit date for the LibraryData object. Returns the last edit date.

```
public string LastEditDate { set; get; }
```

- `MetaData`. Gets or sets the content metadata list for the LibraryData object. Returns the content metadata list.

```
public Ektron.Cms.ContentMetaData[] MetaData { set; get; }
```

- `OriginalLibraryId`. The Id of the original library item. For example, when dealing with a thumbnail image, this is the master image it was generated from. Returns the Id of the original library item.

```
public long OriginalLibraryId { set; get; }
```

- `ParentId`. Gets or sets the parent's ID for the LibraryData object. Returns the parent's long ID.

```
public long ParentId { set; get; }
```

- `Relative`. Gets or sets the Relative for the LibraryData object. Returns the Relative.

```
public int Relative { set; get; }
```

- `Taxonomies`. Gets or sets the taxonomy data for this LibraryData object. Returns the taxonomydata list.

```
public Ektron.Cms.TaxonomyBaseData[] Taxonomies { set; get; }
```

- `Teaser`. Gets or sets the teaser for the LibraryData object. Returns the teaser.

```
public string Teaser { set; get; }
```

- `UserId`. Gets or sets the user's ID for the LibraryData object. Returns the library user's long ID.

```
public long UserId { set; get; }
```

MetadataTypeManager

8.50 and higher

The MetadataTypeManager class manages metadata types, such as title and other artifacts. For information about metadata types and definitions, see Working with Metadata in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Content.MetadataTypeManager
```

Constructors

- `MetadataTypeManager()`
- `MetadataTypeManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `MetadataTypeService`. Returns an instance of the business objects MetadataType Service.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.ContentMetaData)
```

Adds a new metadata type.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Title
- Editable
- Search Allowed
- Default Text
- Seperator

Parameters

- `metadata`. The ContentMetaData object.

Returns

Returns the custom ContentMetaData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxMetaTitle"
CssClass="span-3 last" runat="server" Text="Title:" />
        <ektronUI:TextField ID="uxMetaTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEditableLabel" AssociatedControlID="uxMetaEditable"
CssClass="span-3 last" runat="server" Text="Editable :" />
        <ektronUI:Button DisplayMode="Checkbox" ID="uxMetaEditable" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSearchLabel" AssociatedControlID="uxSearchAllowed"
CssClass="span-3 last" runat="server" Text="Search Allowed :" />
        <ektronUI:Button DisplayMode="Checkbox" ID="uxSearchAllowed" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDefaultTextLabel" AssociatedControlID="uxDefault"
CssClass="span-3 last" runat="server" Text="Default Text :" />
        <ektronUI:TextField ID="uxDefault" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxSeperatorLabel" AssociatedControlID="uxSeperator"
```

```
CssClass="span-3 last" runat="server" Text="Seperator :" />
        <ektronUI:TextField ID="uxSeperator" CssClass="span-1" runat="server"
ValidationGroup="RegisterValidationGroup" Text=";" />
    </li>

    <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MetadataTypeManager metaTypeManager = new MetadataTypeManager();
            ContentMetaData cMetaTypeData = new ContentMetaData()
            {
                Name = uxMetaTitle.Text,
                DataType  =
Ektron.Cms.Common.EkEnumeration.ContentMetadataDataType.Text,
                IsSearchAllowed = uxSearchAllowed.Checked,
                Separator =  uxSeperator.Text ,
                Type =
Ektron.Cms.Common.EkEnumeration.ContentMetadataType.SearchableProperty,
                IsEditable = uxMetaEditable.Checked,
                DefaultValue = uxDefault.Text
            };


            metaTypeManager.Add(cMetaTypeData);

            MessageUtilities.UpdateMessage(uxMessage, "Metadata type Added with Id " +
cMetaTypeData.Id, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a metadata type from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Metadata Type ID

Parameters

- `metadataTypeId`. The ID of the metadata type to be deleted.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxMetaTypeIdLabel" AssociatedControlID="uxMetaTypeId"
CssClass="span-6 last" runat="server" Text="Metadata Type Id:" />
            <ektronUI:TextField ID="uxMetaTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
        </li>

        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long metaTypeID = long.Parse(uxMetaTypeId.Text);

            MetadataTypeManager metaTypeManager = new MetadataTypeManager();
            metaTypeManager.Delete(metaTypeID);

            MessageUtilities.UpdateMessage(uxMessage,  "Metadata Type with Id " +
uxMetaTypeId.Text + " deleted", Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.ContentData)
```

Retrieves the properties of a specific MetadataType object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Metadata Type ID

Parameters

- `metadataTypeId`. The ID of the metadata type to get.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMetaTypeIdLabel" AssociatedControlID="uxMetaTypeId"
CssClass="span-6 last" runat="server" Text="Metadate Type Id :" />
        <ektronUI:TextField ID="uxMetaTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
        <li class="clearfix">
        <asp:Literal ID="uxTypeName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSearchable" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDefault" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSeperator" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long metaDataTypeID = long.Parse(uxMetaTypeId.Text);

            MetadataTypeManager metadataTypeManager = new MetadataTypeManager();
            ContentMetaData metaData = metadataTypeManager.GetItem(metaDataTypeID);

            if (metaData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for Metadata type
with ID " + metaData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxTypeName.Text = "Type Name : " + metaData.Name;
                uxSeperator.Text = "Separator : " + metaData.Separator;
                uxSearchable.Text = "Searchable : " + metaData.IsSearchAllowed;
                uxDefault.Text = "Default Text : " + metaData.DefaultValue;
            }
            else
            {
```

```
                uxMessage.Text = "Metadata type does not exist. ";
                MessageUtilities.UpdateMessage(uxMessage, "Activity",
Message.DisplayModes.Error);

            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Content.MetadataTypeCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Metadata Type Property
- * Object Value

Parameters

- `criteria`. MetadataTypeCriteria used to retrieve content.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMetadataTypePropertyLabel"
AssociatedControlID="uxMetadataTypeProperty" CssClass="span-4 last" runat="server"
Text="* MetadataTypeProperty:" />
        <asp:DropDownList ID="uxMetadataTypeProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
```

```
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxContentMetaDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                        <th>
                        Type Name
                    </th>
                    <th>
                        Language
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("TypeId")%>
            </td>
                <td class="devsite-method">
                <%# Eval("TypeName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Language")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Content;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            MetadataTypeManager metadataTypeManager = new MetadataTypeManager();

            MetadataTypeCriteria criteria = new MetadataTypeCriteria();
            if (uxMetadataTypeProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(MetadataTypeProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(MetadataTypeProperty.Name,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            List<ContentMetaData> contentMetaDataList = metadataTypeManager.GetList
(criteria);

            uxContentMetaDataListView.Visible = true;
            uxContentMetaDataListView.DataSource = contentMetaDataList;
            uxContentMetaDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.ContentMetaData)
```

Updates an existing content metadata item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Metadata Type ID
- Title
- * Editable
- * Search Allowed
- Default Text
- Separator

Parameters

- `metadata`. The ContentMetaData object.

Returns

Returns the custom ContentMetaData object updated.

Remarks

Validate the ContentMetadata item with `GetItem()` before you update the properties. Then, call `Update` with your modified ContentMetaData object.

NOTE: You cannot change all properties after the initial Add event, such as the ContentMetaData.Type.\

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMetaTypeIDLabel" AssociatedControlID="uxMetaTypeID"
CssClass="span-5 last" runat="server" Text="* Metadata Type ID :" />
        <ektronUI:TextField ID="uxMetaTypeID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxMetaTitle"
CssClass="span-5 last" runat="server" Text="Title :" />
        <ektronUI:TextField ID="uxMetaTitle" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEditableLabel" AssociatedControlID="uxMetaEditable"
CssClass="span-5 last" runat="server" Text="* Editable :" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxMetaEditable"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSearchLabel" AssociatedControlID="uxSearchAllowed"
CssClass="span-5 last" runat="server" Text="* Search Allowed :" />
```

```
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxSearchAllowed"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDefaultTextLabel" AssociatedControlID="uxDefault"
CssClass="span-5 last" runat="server" Text="Default Text :" />
        <ektronUI:TextField ID="uxDefault" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSeperatorLabel" AssociatedControlID="uxSeperator"
CssClass="span-5 last" runat="server" Text="Seperator :" />
        <ektronUI:TextField ID="uxSeperator" CssClass="span-1" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long metaDataTypeID = long.Parse(uxMetaTypeID.Text);
            MetadataTypeManager metaTypeManager = new MetadataTypeManager();

            ContentMetaData cMetaTypeData = metaTypeManager.GetItem(metaDataTypeID);

            if (cMetaTypeData != null)
            {
                cMetaTypeData.Name = (uxMetaTitle.Text != string.Empty ?
uxMetaTitle.Text :cMetaTypeData.Name) ;
                cMetaTypeData.IsSearchAllowed = uxSearchAllowed.Checked;
                cMetaTypeData.Separator = (uxSeperator.Text != string.Empty ?
uxSeperator.Text : cMetaTypeData.Separator);
                cMetaTypeData.IsEditable = uxMetaEditable.Checked;
                cMetaTypeData.DefaultValue = (uxDefault.Text != string.Empty ?
```

```
uxDefault.Text : cMetaTypeData.DefaultValue);

                metaTypeManager.Update(cMetaTypeData);

                MessageUtilities.UpdateMessage(uxMessage,  "Metadata type with Id " +
cMetaTypeData.Id + " updated", Message.DisplayModes.Success);

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Metadata type does not
exist.", Message.DisplayModes.Error);

            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

ContentMetaData

Namespace

```
Ektron.Cms
```

Properties

- `AllowMulti`

  ```
  public bool AllowMulti { set; get; }
  ```

- `AllowMultiple`

  ```
  public bool AllowMultiple { set; get; }
  ```

- `BlogMetaType`

  ```
  public int BlogMetaType { set; get; }
  ```

- `CaseSensitive`

  ```
  public bool CaseSensitive { set; get; }
  ```

- `DataType`. Gets or sets the DataType used by this metadata definition.

  ```
  public EkEnumeration.ContentMetadataDataType DataType { set; get; }
  ```

- `DefaultText`

  ```
  public string DefaultText { set; get; }
  ```

- `DefaultValue`. Gets or sets the default value for this metadata definition.

  ```
  public string DefaultValue { set; get; }
  ```

- `Editable`

```
public bool Editable { set; get; }
```

- `IsDisplayable`. Gets or sets the Displayable flag.

```
public bool IsDisplayable { set; get; }
```

- `IsEditable`. Gets or sets the Editable flag.

```
public bool IsEditable { set; get; }
```

- `IsMetadataBlog`

```
public bool IsMetadataBlog { get; }
```

- `IsSearchAllowed`. Gets or sets the SearchAllowed flag.

```
public bool IsSearchAllowed { set; get; }
```

- `IsSelectableOnly`

```
public bool IsSelectableOnly { set; get; }
```

- `Language`

```
public int Language { set; get; }
```

- `MetaDataObjectType`. Type of object to which this metadata applies.

```
public EkEnumeration.MetaDataType MetaDataObjectType { set; get; }
```

- `MetaDisplay`

```
public bool MetaDisplay { set; get; }
```

- `MetaDisplayEE`

```
public bool MetaDisplayEE { set; get; }
```

- `Name`

```
public string Name { set; get; }
```

- `ObjectType`

```
public int ObjectType { set; get; }
```

- `RemoveDuplicate`

```
public bool RemoveDuplicate { set; get; }
```

- `Required`

```
public bool Required { set; get; }
```

- `SearchAllowed`

```
public bool SearchAllowed { set; get; }
```

- `SelectableText`

```
public string SelectableText { set; get; }
```

- `Separater`

```
public string Separater { set; get; }
```

- `Separator`

```
public string Separator { set; get; }
```

- `TagType`. Tag Type:
  - 0=EkConstants.MetaTagType_Html
  - 1=EkConstants.MetaTagType_Meta

- 2=EkConstants.MetaTagType_Collection
- 3=EkConstants.MetaTagType_ListSummary
- 4=EkContansts.MetaTagType_Content
- 5=EkConstants.MetaTagType_Image
- 6=EkConstants.MetaTagType_HyperLink
- 7=EkConstants.MetaTagType_File
- 8=EkConstants.MetaTagType_Menu
- 9=EkConstants.MetaTagType_User
- 100=EkConstants.MetaTagType_Searchable

```
public int TagType { set; get; }
```

- `Text`

```
public string Text { set; get; }
```

- `Title`

```
public string Title { set; get; }
```

- `Type`. The type of the metadata definition

```
public EkEnumeration.ContentMetadataType Type { set; get; }
```

- `TypeId`

```
public long TypeId { set; get; }
```

- `TypeName`

```
public string TypeName { set; get; }
```

- `Validate()`. Validate this instance.

```
public override ValidationResults Validate()
```

MetadataTypeCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Content
```

Properties

- `MetadataTypeCriteria()`

```
public MetadataTypeCriteria()
```

- `MetadataTypeCriteria(Ektron.Cms.Content.MetadataTypeProperty, EkEnumeration.OrderByDirection)`

```
public MetadataTypeCriteria(Ektron.Cms.Content.MetadataTypeProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `MetadataTypeProperty` are:

- Id
- Language
- Name

- ObjectType
- Required

TemplateManager

8.50 and higher

The TemplateManager class manages templates, which typically includes page headers, footers, and placeholders for content, forms, summaries, calendars, collections, or other page elements. For information about templates, see Working with Templates in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Content.TemplateManager
```

Constructors

- `TemplateManager()`
- `TemplateManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `TemplateService`. Returns an instance of the business objects Template Service.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.TemplateData)
```

Adds a template based on information in a TemplateData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Template Name
- Description
- * Template File

Parameters

- `templateData`. The TemplateData object to add.

Returns

TemplateData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTemplateNameLabel" AssociatedControlID="uxTemplateName"
CssClass="span-3 last"
            runat="server" Text="*Template Name:" />
        <ektronUI:TextField ID="uxTemplateName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-3 last"
            runat="server" Text="Description :" />
        <ektronUI:TextField ID="uxDescription" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <asp:Label ID="uxFilePathLabel" AssociatedControlID="uxFilePath" CssClass="span-
3 last"
            runat="server" Text="*Template File :(Choose Template from Root of your
website of Type'.aspx'):" />
        <asp:FileUpload ID="uxFilePath" size="29.75" CssClass="span-4" runat="server" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
```

```
Text="Add"></ektronUI:Button>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TemplateManager templateManager = new TemplateManager();
            string filename = uxFilePath.PostedFile.FileName.ToString();
            if (!(filename.Contains(".aspx")))
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Upload file with type
'.aspx' {page} Only.", Message.DisplayModes.Error);
                return;
            }
            TemplateData templateData = new TemplateData()
            {
                TemplateName = uxTemplateName.Text,
                Description = uxDescription.Text,
                Type = Ektron.Cms.Common.EkEnumeration.TemplateType.Default,
                FileName = filename,
            };

            templateManager.Add(templateData);

            if (templateData.Id > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Template Added with Id " +
templateData.Id, Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Template already exists in the
system.", Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
```

```
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a template from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Template ID

Parameters

- `templateId`. The ID of the template to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxTemplateIdLabel" AssociatedControlID="uxTemplateId"
CssClass="span-3 last"
                runat="server" Text="*Template Id:" />
            <ektronUI:TextField ID="uxTemplateId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
        </li>
        <li class="clearfix" style="color: red;">
            <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                Text="Status:" />
        </li>
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    </ol>

    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

```
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
<ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxTemplateIdLabel" AssociatedControlID="uxTemplateId"
CssClass="span-3 last"
                runat="server" Text="*Template Id:" />
            <ektronUI:TextField ID="uxTemplateId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
        </li>
        <li class="clearfix" style="color: red;">
            <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                Text="Status:" />
        </li>
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    </ol>

    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TemplateManager templateManager = new TemplateManager();
            long templateId = long.Parse(uxTemplateId.Text);
            TemplateData templateData = templateManager.GetItem(templateId);
            if (templateData != null)
            {
                templateManager.Delete(templateId);
                MessageUtilities.UpdateMessage(uxMessage, "Template with ID " +
templateId + " has been Deleted.", Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Template
Id.", Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.TemplateData)
```

Retrieves the properties of a specific TemplateData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Template ID

Parameters

- `templateId`. The ID of the template to retrieve.

Returns

TemplateData object found.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTemplateIdLabel" AssociatedControlID="uxTemplateId"
CssClass="span-3 last"
            runat="server" Text="*Template Id:" />
        <ektronUI:TextField ID="uxTemplateId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
</ol>

    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
        <li class="clearfix">
        <asp:Literal ID="uxTemplateName" runat="server"></asp:Literal>
    </li>
```

```
    <li class="clearfix">
        <asp:Literal ID="uxTemplateDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFileName" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TemplateManager templateManager = new TemplateManager();
            long templateId = long.Parse(uxTemplateId.Text);
            TemplateData templateData = templateManager.GetItem(templateId);
            if (templateData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for Template with ID
" + templateId + " are below", Message.DisplayModes.Success);

                uxTemplateName.Text = "Template Name : " + templateData.TemplateName;
                uxTemplateDescription.Text = "Template Description : " +
templateData.Description;
                uxFileName.Text = "FileName : " + templateData.FileName;
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Template
Id.", Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Common.TemplateCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Template Property
- * Object Value

Parameters

- `criteria`. Object used to perform filtering based on several TemplateData object properties.

Returns

A list of TemplateData objects found.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTemplatePropertyLabel"
AssociatedControlID="uxTemplateProperty" CssClass="span-4 last" runat="server"
Text="*TemplateProperty : " />
        <asp:DropDownList ID="uxTemplateProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>TemplateName</asp:ListItem>
            <asp:ListItem>FileName</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxTemplateDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                        <th>
                        FileName
                    </th>
                    <th>
                        TemplateName
                    </th>
                    <th>
                        Description
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
                <td class="devsite-method">
                <%# Eval("FileName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TemplateName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Description")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
```

```
using Ektron.Cms.Common;
using Ektron.Cms.Content;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TemplateManager templateManager = new TemplateManager();
            object Objectvalue;

            TemplateCriteria criteria = new TemplateCriteria();
            if (uxTemplateProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(TemplateDataProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (uxTemplateProperty.SelectedItem.Text == "TemplateName")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TemplateDataProperty.TemplateName,
CriteriaFilterOperator.Contains, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TemplateDataProperty.FileName,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            List<TemplateData> TemplateDataList = templateManager.GetList(criteria);

            uxTemplateDataListView.Visible = true;
            uxTemplateDataListView.DataSource = TemplateDataList;
            uxTemplateDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.TemplateData)
```

Updates an existing Template item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Template ID
- Template Name
- Description
- Template File

Parameters

- `templateData`. The TemplateData object to update.

Returns

Returns the custom TemplateData object updated.

Remarks

Validate the Template item with `GetItem()` before you update the properties. Then, call `Update` with your modified TemplateData object.

NOTE: You cannot change all properties after the initial Add event, such as the TemplateData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTemplateIdLabel" AssociatedControlID="uxTemplateId"
CssClass="span-3 last"
            runat="server" Text="*Template Id:" />
        <ektronUI:TextField ID="uxTemplateId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTemplateNameLabel" AssociatedControlID="uxTemplateName"
CssClass="span-3 last"
            runat="server" Text="Template Name:" />
        <ektronUI:TextField ID="uxTemplateName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-3 last"
            runat="server" Text="Description :" />
        <ektronUI:TextField ID="uxDescription" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <asp:Label ID="uxFilePathLabel" AssociatedControlID="uxFilePath" CssClass="span-
3 last"
            runat="server" Text="Template File :" />
        <asp:FileUpload ID="uxFilePath" size="29.75" CssClass="span-4" runat="server" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TemplateManager templateManager = new TemplateManager();
            long templateId = long.Parse(uxTemplateId.Text);
            TemplateData templateData = templateManager.GetItem(templateId);
            string filename = uxFilePath.PostedFile.FileName.ToString();

            if (templateData != null)
            {
                if (!string.IsNullOrEmpty(filename))
                {
                    if (!(filename.Contains(".aspx")))
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Upload file
with type '.aspx' {page} Only.", Message.DisplayModes.Error);
                        return;
                    }
                }

                templateData.TemplateName = uxTemplateName.Text != string.Empty ?
uxTemplateName.Text : templateData.TemplateName;
                templateData.Description = uxDescription.Text != string.Empty ?
uxDescription.Text : templateData.Description;
```

```
                templateData.FileName = uxFilePath.PostedFile.FileName.ToString() !=
string.Empty ? uxFilePath.PostedFile.FileName.ToString() : templateData.FileName;

                templateManager.Update(templateData);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus,"Please Enter valid Template
Id.", Message.DisplayModes.Error);
                return;
            }
            if (templateData.Id > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Template with Id " +
templateData.Id + " has updated", Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Template already exists in the
system.", Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

TemplateData

Namespace

```
Ektron.Cms
```

Properties

- Description

  ```
  public string Description { set; get; }
  ```

- FileName

  ```
  public string FileName { set; get; }
  ```

- MasterLayoutID

  ```
  public long MasterLayoutID { set; get; }
  ```

- SubType

```
public Ektron.Cms.Controls.CmsWebService.TemplateSubType
   SubType { set; get; }
```

- TemplateName

```
public string TemplateName { set; get; }
```

- Thumbnail

```
public string Thumbnail { set; get; }
```

- Type

```
public Ektron.Cms.Controls.CmsWebService.TemplateType
  Type { set; get; }
```

Flag

8.60 and higher

The Flag manager category manages a means by which an end user provides feedback about content (because the content is offensive, or because the user likes the content). The feedback goes to an administrator or moderator for action. It has the following class.

- *FlagDefinitionManager* on the facing page

FlagDefinitionManager

8.60 and higher

The class manages a means by which an end user provides feedback about content (because the content is offensive, or because the user likes the content). The feedback goes to an administrator or moderator for action.

Namespace

```
Ektron.Cms.Framework.Flag.FlagDefinitionManager
```

Constructors

- `FlagDefinitionManager()`
- `FlagDefinitionManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `FlagDefinitionManagerService`. Returns an instance of the business objects flag definition manager service.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 1189
- *GetItem* on page 1191
- *GetList* on page 1193
- *Update* on page 1196

Add

```
Add(Ektron.Cms.FlagDefData)
```

Adds a flag based on information in FlagDefData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- Description
- * Flag Item Title
- Flag Item Description

Parameters

- `FlagDefinitionData`. The flag object to add.

Returns

FlagDefData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
       <asp:Label ID="uxFlagDefinitionNameLabel"
AssociatedControlID="uxFlagDefintionName" CssClass="span-4 last" runat="server" Text="*
Title:" />
       <asp:TextField ID="uxFlagDefintionName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30"  />
    </li>
    <li class="clearfix">
       <asp:Label ID="uxFlagDefinitionDescriptionLabel"
AssociatedControlID="uxFlagDefinitionDescription" CssClass="span-4 last" runat="server"
Text=" Description:" />
       <asp:TextField ID="uxFlagDefinitionDescription" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30"  />
    </li>
   <li class="clearfix">
       <asp:Label ID="uxFlagItemNameLabel" AssociatedControlID="uxFlagItemName"
CssClass="span-4 last" runat="server" Text="* Flag Item Title:" />
       <asp:TextField ID="uxFlagItemName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30"  />
    </li>
    <li class="clearfix">
       <asp:Label ID="uxFlagItemDescriptionLabel"
AssociatedControlID="uxFlagItemDescription" CssClass="span-4 last" runat="server" Text="
Item Description:" />
       <asp:TextField ID="uxFlagItemDescription" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30"  />
    <li class="clearfix">
```

```
        <asp:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <asp:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System;
using System.Collections.Generic;
using Ektron.Cms;
using Ektron.Cms.Framework.Flag ;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            FlagDefinitionManager flagDefinitionManager=new FlagDefinitionManager();
            List<FlagItemData> flagItemList = new List<FlagItemData>();
            FlagItemData flagItemData = new FlagItemData()
            {
                 Name = uxFlagItemName.Text,
                 Description =uxFlagItemDescription.Text
            };
            flagItemList.Add(flagItemData);

            FlagDefData flagDefinitionData = new FlagDefData()
            {
                  Name =uxFlagDefintionName.Text,
                  Description = uxFlagDefinitionDescription.Text,
                  Items = flagItemList.ToArray()
            };

            flagDefinitionManager.Add(flagDefinitionData);
            MessageUtilities.UpdateMessage(uxMessage, "Flag Definition with ID " +
flagDefinitionData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a flag from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The ID of the flag to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFlagDefinitionIdLabel"
AssociatedControlID="uxFlagDefinitionId" CssClass="span-4 last" runat="server" Text="*
Flag Definition Id:" />
        <ektronUI:TextField ID="uxFlagDefinitionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms.Framework.Flag;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            long flagDefinitionId = long.Parse(uxFlagDefinitionId.Text);
            FlagDefinitionManager flagDefinitionManager = new FlagDefinitionManager();
            FlagDefData flagDefinitionData = flagDefinitionManager.GetItem
(flagDefinitionId);
            if (flagDefinitionData != null)
            {
                flagDefinitionManager.Delete(flagDefinitionData.ID);
                MessageUtilities.UpdateMessage(uxMessage, "Flag Definition deleted.",
Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus,  "Please Enter Valid Flag ID.",
Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific FlagDefData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `FlagDefinitionId`. The ID of the flag to retrieve.

Returns

FlagDefData object found.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFlagDefinitionIdLabel"
AssociatedControlID="uxFlagDefinitionId" CssClass="span-3 last" runat="server" Text="*
Flag Definition Id :" />
        <ektronUI:TextField ID="uxFlagDefinitionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFlagDefinitionName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFlagDefinitionDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFlagItemTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFlagItemDescription" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms;
using Ektron.Cms.Framework.Flag;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long flagDefinitionId = long.Parse(uxFlagDefinitionId.Text);
            FlagDefinitionManager flagDefinitionManager = new FlagDefinitionManager();
            FlagDefData flagDefinitionData = flagDefinitionManager.GetItem
(flagDefinitionId);
            if (flagDefinitionData != null)
            {
                uxId.Text = "FlagDefinition Id : " + flagDefinitionData.Id.ToString();
                uxFlagDefinitionName.Text = "FlagDefinition Name : " +
flagDefinitionData.Name;
                uxFlagDefinitionDescription.Text = "FlagDefinition Description : " +
flagDefinitionData.Description;
                foreach (FlagItemData flagItemData in flagDefinitionData.Items)
                {
                    uxFlagItemTitle.Text = "Flag Item Title : " + flagItemData.Name;
                    uxFlagItemDescription.Text = "Flag Item Description : " +
flagItemData.Description;
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Flag Definition does not
exists. ", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Flag.FlagDefinitionCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Flag Definition Data Property
- * Object Value

Parameters

- `criteria`. Object used to perform filtering based on several FlagDefinitionCriteria object properties.

Returns

A list of FlagDefData objects found.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFlagDefinitionDataPropertyLabel"
AssociatedControlID="uxFlagDefinitionDataProperty" CssClass="span-6 last" runat="server"
Text="Flag Definition Data Property:" />
        <asp:DropDownList ID="uxFlagDefinitionDataProperty" runat="server">
            <asp:ListItem>Flag Entry Id</asp:ListItem>
            <asp:ListItem>Object Id</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFlagDefinitionValueLabel"
AssociatedControlID="uxFlagDefinitionValue" CssClass="span-6 last" runat="server"
Text="* Object Value:" />
        <ektronUI:TextField ID="uxFlagDefinitionValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxFlagDefinitionListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Flag Definition Id
```

```
                    </th>
                    <th>
                        Flag Definition Name
                    </th>
                    <th>
                        Flag Definition Description
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("ID")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Description")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Flag;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            Ektron.Cms.Flag.FlagDefinitionCriteria criteria = new
Ektron.Cms.Flag.FlagDefinitionCriteria();
            if (uxFlagDefinitionDataProperty.SelectedItem.Text == "Flag Definition Id")
            {
                Objectvalue = Convert.ToInt64(uxFlagDefinitionValue.Text);
                criteria.AddFilter( FlagDefinitionProperty.Id,
```

```
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxFlagDefinitionValue.Text;
                criteria.AddFilter(FlagDefinitionProperty.FlagDefName  ,
CriteriaFilterOperator.Contains , Objectvalue);
            }

            FlagDefinitionManager flagDefinitionManager = new FlagDefinitionManager();
            List<FlagDefData> flagDefinitionList = flagDefinitionManager.GetList
(criteria);

            uxFlagDefinitionListView.Visible = true;
            uxFlagDefinitionListView.DataSource = flagDefinitionList;
            uxFlagDefinitionListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.FlagDefData)
```

Updates an existing flag item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Title
- * Item ID
- * Item Title

Parameters

- `FlagDefinitionData`. The FlagDefData object to update.

Returns

Returns the FlagDefData object updated.

Remarks

Validate the FlagDefData item with `GetItem()` before you update the properties. Then, call `Update` with your modified FlagDefData object.

NOTE: You cannot change all properties after the initial Add event.

.aspx code snippet

```
<ol class="formFields">
      <li class="clearfix">
       <ektronUI:Label ID="uxFlagDefinitionIdLabel"
AssociatedControlID="uxFlagDefintionId" CssClass="span-4 last" runat="server" Text="*
ID:" />
       <ektronUI:TextField ID="uxFlagDefintionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFlagDefinitionNameLabel"
AssociatedControlID="uxFlagDefintionName" CssClass="span-4 last" runat="server" Text="*
Title:" />
        <ektronUI:TextField ID="uxFlagDefintionName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30"  />
    </li>
   <li class="clearfix">
       <ektronUI:Label ID="uxFlagItemIdLabel" AssociatedControlID="uxFlagItemId"
CssClass="span-4 last" runat="server" Text="* Item ID:" />
       <ektronUI:TextField ID="uxFlagItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFlagItemNameLabel" AssociatedControlID="uxFlagItemName"
CssClass="span-4 last" runat="server" Text="* Item Title:" />
        <ektronUI:TextField ID="uxFlagItemName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30"  />
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Community;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long flagDefinitionId = long.Parse(uxFlagDefintionId.Text);
            FlagDefinitionManager flagDefinitionManager = new FlagDefinitionManager();

            FlagDefData flagDefinitionData = flagDefinitionManager.GetItem
(flagDefinitionId);
            FlagItemData flagItemData = new FlagItemData()
            {

                 ID = long.Parse(uxFlagItemId.Text),
                 Name  =uxFlagItemName.Text
            };
            List<FlagItemData> flagItemList = new List<FlagItemData>();
            flagItemList.Add(flagItemData);


            if (flagDefinitionData != null)
            {

                flagDefinitionData.Name =uxFlagDefinitionName.Text;
                flagDefinitionData.Items = flagItemList.ToArray();
                flagDefinitionManager.Update(flagDefinitionData);

                MessageUtilities.UpdateMessage(uxMessage, "Flag Definition Updated ",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Flag Definition not exists.
", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

FlagDefData

Namespace

```
Ektron.Cms
```

Properties

- `Description`

```
public string Description { set; get; }
```

- `Hidden`

```
public bool Hidden { set; get; }
```

- `ID`

```
public long ID { set; get; }
```

- `Items`

```
public Ektron.Cms.Controls.CmsWebService.FlagItemData[] Items { set; get; }
```

- `Language`

```
public int Language { set; get; }
```

- `m_bHidden`

```
public bool m_bHidden { set; get; }
```

- `m_strDescription`

```
public string m_strDescription { set; get; }
```

- `m_strName`

```
public string m_strName { set; get; }
```

- `Name`

```
public string Name { set; get; }
```

FlagDefinitionCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Flag
```

Properties

- `FlagDefinitionCriteria()`

```
public FlagDefinitionCriteria()
```

- `FlagDefinitionCriteria(Ektron.Cms.Common.FlagDefProperty, EkEnumeration.OrderByDirection)`

```
public FlagDefinitionCriteria(Ektron.Cms.Common.FlagDefProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

  Accepted fields for the `FlagDefinitionProperty` are:

  - FlagDefDescription
  - FlagDefLanguage
  - FlagDefName
  - FlagLastEditDate
  - Id
  - IsCommunityFlag

Notifications

8.50 and higher

The Notifications manager category manages information that is sent to users and has the following class:

- *NotificationManager* below

NotificationManager

8.50 and higher

The NotificationManager class manages data that is sent to users.

Namespace

```
Ektron.Cms.Framework.Notifications.NotificationManager
```

Constructors

- `NotificationManager()`
- `NotificationManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `IsNotificationReplyMessagePrepended`. Gets setting indicating if the Notification Email Reply message should be prepended to the notification email.
- `NotificationService`. Notification Service
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Send (activity data)* below
- *Send (message data)* on page 1203

Send (activity data)

```
Send(Ektron.Cms.Activity.ActivityData)
```

Sends notifications based upon the supplied activity.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID
- * Activity Message

Parameters

- `activity`. The ActivityData object that has occurred and will result in notifications being sent. The Activity.Message property is ignored and is retrieved based upon CMS message settings.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIDLabel" AssociatedControlID="uxContentID"
CssClass="span-4 last"
            runat="server" Text="*Content ID :" />
        <ektronUI:TextField ID="uxContentID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="30" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMesageLabel" AssociatedControlID="uxActivityMessage"
CssClass="span-4 last"
            runat="server" Text="*Activity Message :" />
        <ektronUI:TextField ID="uxActivityMessage" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="A New Content is added" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Send"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Notifications;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationManager notificationManager = new NotificationManager();
            ActivityManager activityManager = new ActivityManager();
            ContentManager contentManager = new ContentManager();
            long contentid = Convert.ToInt64(uxContentID.Text);
            //Check the Content Details
            ContentData contentData = contentManager.GetItem(contentid);
            if (contentData != null)
            {
                ActivityUserInfo userInfo = new ActivityUserInfo();
                userInfo.Id = contentData.UserId;
                //Set the Activity Details
                ActivityData activityData = new ActivityData()
                {
                    ActivityTypeId = 0,
                    ActionUser = userInfo,
                    Message = uxActivityMessage.Text,
                    ObjectId = contentid,
                    Date = DateTime.Now,
                    LanguageId = 1033
                };
                //Add a New Activity with given activityDetails.
                activityManager.Add(activityData);
                //Send notification based upon the supplied activity.
                notificationManager.Send(activityData);
                MessageUtilities.UpdateMessage(uxMessage, "Successfully Sent a
Notification Based upon the Supplied Activity.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Content ID
and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

```
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Send (message data)

```
Send(Ektron.Cms.Notifications.NotificationMessageData)
```

Sends notifications based upon the supplied notification message data.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID
- * Subject
- * Activity Message
- * User ID

Parameters

- `notificationMessage`. NotificationMessageData object defining message to send and the user to whom to send it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIDLabel" AssociatedControlID="uxContentID"
CssClass="span-4 last"
            runat="server" Text="*Content ID :" />
        <ektronUI:TextField ID="uxContentID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="30" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxSubjectLabel" AssociatedControlID="uxSubject"
CssClass="span-4 last"
            runat="server" Text="*Subject :" />
        <ektronUI:TextField ID="uxSubject" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="New ontrek Notification" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxMesageLabel" AssociatedControlID="uxActivityMessage"
CssClass="span-4 last"
            runat="server" Text="*Activity Message :" />
        <ektronUI:TextField ID="uxActivityMessage" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="A New Content is added" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Send"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Notifications;
using Ektron.Cms.Activity;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Notifications;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationManager notificationManager = new NotificationManager();
            ActivityManager activityManager = new ActivityManager();

            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxUserId.Text);

            ContentManager contentManager = new ContentManager();
            long contentid = Convert.ToInt64(uxContentID.Text);
            //Check the Content Details
```

```
            ContentData contentData = contentManager.GetItem(contentid);
            if (contentData != null)
            {   //Check Whether Entered UserID valid or Not
                UserData Userdata = Usermanager.GetItem(UserId);
                if (Userdata != null)
                {
                    ActivityUserInfo userInfo = new ActivityUserInfo();
                    userInfo.Id = contentData.UserId;
                    //Set the Activity Details
                    ActivityData activityData = new ActivityData()
                    {
                        ActivityTypeId = 0,
                        ActionUser = userInfo,
                        Message = uxActivityMessage.Text,
                        ObjectId = contentid,
                        Date = DateTime.Now,
                        LanguageId = 1033
                    };
                    //Add a New Activity with given activityDetails.
                    activityManager.Add(activityData);
                    UserNotificationData userNotificationData = new UserNotificationData
();
                    userNotificationData.User = Userdata;

                    //AgentName is a Default value which is taken from the web.config
file under notificationAgent/providers section
                    userNotificationData.AgentName = "EktronEmail";
                    string message = uxActivityMessage.Text + "<p> Content ID is " +
contentData.Id + "</p><p><a href=http://" + Request.Headers["host"] +
contentData.Quicklink + ">" + contentData.Title + "</a> </p>";
                    NotificationMessageData notificationMessageData = new
NotificationMessageData()
                    {
                        HtmlMessage = message,
                        Subject = uxSubject.Text,
                        UserNotification = userNotificationData,
                        TextMessage = uxActivityMessage.Text,
                        DocumentTitle = contentData.Title,
                        Activity = activityData,
                        Document = null,
                    };
                    //Send notification based upon the supplied notification Message
Data.
                    notificationManager.Send(notificationMessageData);
                    MessageUtilities.UpdateMessage(uxMessage, "Successfully Sent a
Notification Based upon the Supplied Activity.", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid User
ID and Try again.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Content ID
```

```
and Try again.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Data Classes

ActivityData

Namespace

```
Ektron.Cms.Activity
```

Properties

- `ActionUser`. Gets or sets the user who caused this activity to occur. For example, the user who posted a blog post or uploaded a document.

  ```
  public Ektron.Cms.Activity.ActivityUserInfo ActionUser { set; get; }
  ```

- `ActivityData()`

  ```
  public ActivityData()
  ```

- `ActivityData(long, long, long)`

  ```
  public ActivityData(long activityTypeId, long objectId,
    long actionUserId)
  ```

- `ActivityData(long, long, long, int, string)`

  ```
  public ActivityData(long activityTypeId, long objectId,
    long actionUserId, int languageId, string message)
  ```

- `ActivityTypeId`. Gets or sets the ActivityType for this Activity. Built in CMS Activity are defined in `EkEnumeration.ActivityType`.

  ```
  public long ActivityTypeId { set; get; }
  ```

- `CanComment`. Gets or sets the CanComment permissions for the activity.

  ```
  public bool CanComment { set; get; }
  ```

- `CommentCount`. Gets or sets the Comment count for the activity.

  ```
  public int CommentCount { set; get; }
  ```

- `Comments`. Gets or sets the Comments for the activity. This property will not be populated unless explicitly specified when calling API. To get the total comment count, see `CommentCount` property.

  ```
  public System.Collections.ObjectModel.Collection<ActivityCommentData>
    Comments { set; get; }
  ```

- `Date`. Gets or sets the date the activity occurred.

```
public System.DateTime Date { set; get; }
```

- `Id`. Gets or set the ID of the activity.

```
public long Id { set; get; }
```

- `LanguageId`. Gets or sets the language associated with this activity's message.

```
public int LanguageId { set; get; }
```

- `Message`. Gets or sets the message for this activity.

```
public string Message { set; get; }
```

- `ObjectId`. Gets or sets the ID of the object associated with this activity.

```
public long ObjectId { set; get; }
```

- `ScopeObjectId`. Gets or sets the ID of the scope object associated with this activity.

```
public long ScopeObjectId { set; get; }
```

- `SiteId`. Gets or sets the ID of the site that the activity originated from.

```
public long SiteId { set; get; }
```

- `TimeLapse`. Gets the time lapse for the activity.

```
public string TimeLapse { set; get; }
```

- `UniqueId`. Gets or sets the unique ID for the activity.

```
public System.Guid UniqueId { set; get; }
```

- `Validate()`. Validates the current ActivityData instance is in a valid state for saving. Returns collection of validation errors if not valid.

```
public ValidationResults Validate()
```

NotificationMessageData

Namespace

```
Ektron.Cms.Notifications
```

Properties

- `Activity`. Gets or sets the Activity associated with the notification. This may be required for some notifications, but not all. The ActivityFeed, for one, provider requires it.

```
public Ektron.Cms.Activity.ActivityData Activity { set; get; }
```

- `Document`. Gets or sets the document associated with the notification message. May be null.

```
public byte[] Document { set; get; }
```

- `DocumentTitle`. Gets or sets the document title associated with the notification message. May be null.

```
public string DocumentTitle { set; get; }
```

- `HtmlMessage`. Gets or sets the HTML message of the notification. This message would be used by providers that support HTML based messages.

```
public string HtmlMessage { set; get; }
```

- `NotificationMessageData()`

```
public NotificationMessageData()
```

- `NotificationMessageData(string, string, string, Ektron.Cms.Notifications.UserNotificationData)`

```
public NotificationMessageData(string htmlMessage,
  string textMessage, string subject,
  Ektron.Cms.Notifications.UserNotificationData userNotification)
```

- `NotificationMessageData(string, string, string, Ektron.Cms.Notifications.UserNotificationData, Ektron.Cms.Activity.ActivityData)`

```
public NotificationMessageData(string htmlMessage,
  string textMessage, string subject,
  Ektron.Cms.Notifications.UserNotificationData
  userNotification, Ektron.Cms.Activity.ActivityData activity)
```

- `Subject`. Gets or sets the subject of the notification.

```
public string Subject { set; get; }
```

- `TextMessage`. Gets or sets the text message of the notification. This message would be used by providers that send messages in plain text and not HTML.

```
public string TextMessage { set; get; }
```

- `UserNotification`. Gets or sets the UserNotificationData for this notification.

```
public Ektron.Cms.Notifications.UserNotificationData
  UserNotification { set; get; }
```

Organization

8.50 and higher

The Organization manager category manages things that help organize content and has the following classes:

- *CollectionManager* on the next page. Manages a list of content.
- *FolderManager* on page 1231. Manages Ektron CMS folders.
- *MenuManager* on page 1263. Manages drop down menus for your Web site.
- *TaxonomyItemManager* on page 1294. Manages taxonomy items.
- *TaxonomyManager* on page 1310. Manages taxonomies.

CollectionManager

8.50 and higher

The CollectionManager class manages a list of content. Information about collections is at Working with Collections in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Organization.CollectionManager
```

Constructors

- `CollectionManager()`
- `CollectionManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `CollectionManagerService`. Returns an instance of the business objects task manager service.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.ContentCollectionData)
```

Adds a content collection based on information in an ContentCollectionData object.

Authenticated users

- CMS Administrators
- Collection Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- * Folder ID
- Recursive
- Summary

Parameters

- `contentCollectionData`. The ContentCollectionData object to add.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCollectionTitleLabel"
AssociatedControlID="uxCollectionTitle" CssClass="span-3 last" runat="server" Text="*
Title:" />
        <ektronUI:TextField  ID="uxCollectionTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last" runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRecursiveLabel" AssociatedControlID="uxRecursive"
CssClass="span-3 last" runat="server" Text="Recursive:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxRecursive" CssClass="span-
6" runat="server"  ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
```

```
        <ektronUI:Label ID="uxCollectionDescriptionLabel"
AssociatedControlID="uxCollectionDescription" CssClass="span-3 last" runat="server"
Text="Summary:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxCollectionDescription"
CssClass="span-4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            CollectionManager collectionManager = new CollectionManager();
            ContentCollectionData contentcollData = new ContentCollectionData()
                {
                    FolderId = long.Parse(uxFolderId.Text),
                    Title = uxCollectionTitle.Text ,
                    Description = uxCollectionDescription.Text,
                    Recursive = uxRecursive.Checked
                };

            collectionManager.Add(contentcollData);

            MessageUtilities.UpdateMessage(uxMessage, "Collection Added with Id " +
contentcollData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

AddContent

```
AddContent(System.Int64,System.Int64)
```

Adds a content item to a collection.

Authenticated users

- CMS Administrators
- Collection Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Collection ID
- * Content ID

Parameters

- `collectionId`. The ID of the content collection.
- `contentId`. The ID of the content.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCollectionIdLabel" AssociatedControlID="uxCollectionId"
CssClass="span-3 last" runat="server" Text="* Collection Id:" />
        <ektronUI:TextField  ID="uxCollectionId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0"  />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0"  />
    </li>
        <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Add
Content"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Framework.Content;
using Ektron.Cms;
using Ektron.Cms.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long collectionId = long.Parse(uxCollectionId.Text);
            long contentId = long.Parse(uxContentId.Text);

            CollectionManager collectionManager = new CollectionManager();
            ContentCollectionData collectionData = collectionManager.GetItem
(collectionId);

            if (collectionData != null)
            {
                ContentManager contentManager = new ContentManager();
                ContentData cData = contentManager.GetItem(contentId, false);
                if (cData != null && cData.Id > 0)
                {
                    collectionManager.AddContent(collectionId, contentId);
                    MessageUtilities.UpdateMessage(uxMessage, "Content Added to
collection", Message.DisplayModes.Success);
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Content
Id.", Message.DisplayModes.Error);
                    return;
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Collection
Id.", Message.DisplayModes.Error);
                return;
            }
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a content collection from the CMS.

Authenticated users

- CMS Administrators
- Collection Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The ID of the content collection to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCollectionIdLabel" AssociatedControlID="uxCollectionId"
CssClass="span-3 last" runat="server" Text="* Collection Id:" />
        <ektronUI:TextField  ID="uxCollectionId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
```

```
runat="server"
                    Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete Content"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
using Ektron.Cms.Framework.Content;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {

            long collectionID = long.Parse(uxCollectionId.Text);

            CollectionManager collectionManager = new CollectionManager();
            ContentCollectionData collectionData = collectionManager.GetItem
(collectionID);
            //Check Whether Entered  CollectionId is Valid or Not.
            if (collectionData != null)
            {
                collectionManager.Delete(collectionID);
                MessageUtilities.UpdateMessage(uxMessage, "Collection with Id " +
uxCollectionId.Text + " deleted", Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Collection does not exist.",
Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeleteContent

```
DeleteContent(System.Int64,System.Int64)
```

Deletes a content item from a collection.

Authenticated users

- CMS Administrators
- Collection Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Collection ID
- * Content ID

Parameters

- `collectionId`. The ID of the content collection.
- `contentId`. The ID of the content.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCollectionIdLabel" AssociatedControlID="uxCollectionId"
CssClass="span-3 last" runat="server" Text="* Collection Id:" />
        <ektronUI:TextField  ID="uxCollectionId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete Content"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
using Ektron.Cms.Framework.Content;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {

            long collectionId = long.Parse(uxCollectionId.Text);
            long contentId = long.Parse(uxContentId.Text);

            CollectionManager collectionManager = new CollectionManager();
            ContentCollectionData collectionData = collectionManager.GetItem
(collectionId);
            //Check Whether Entered  CollectionId is Valid or Not.
            if (collectionData != null)
            {
                ContentManager contentManager = new ContentManager();
                ContentData cData = new ContentData();
                cData = contentManager.GetItem(contentId, false);
                //Check Whether Entered  ContentId is Valid or Not.
                if (cData != null)
                {
                    collectionManager.DeleteContent(collectionId, contentId);
                    MessageUtilities.UpdateMessage(uxMessage, "Content Id deleted from
collection", Message.DisplayModes.Success);
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
ContentID.", Message.DisplayModes.Error);
                    return;
```

```
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Collection does not exist.",
Message.DisplayModes.Error);
                return;
            }


            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.ContentCollectionData)
```

Retrieves the properties of a specific ContentCollectionData object.

Authenticated users

- CMS Administrators
- Collection Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The ID of the content collection to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCollectionIdLabel" AssociatedControlID="uxCollectionId"
CssClass="span-4 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxCollectionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFolderId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxRecursive" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTemplate" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long collectionID = long.Parse(uxCollectionId.Text);

            CollectionManager collectionManager = new CollectionManager();
            ContentCollectionData collectionData = collectionManager.GetItem
(collectionID);

            if (collectionData != null)
            {

                MessageUtilities.UpdateMessage(uxMessage, "Details for Collection with
```

```
ID " + collectionData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxTitle.Text = "Title : " + collectionData.Title;
                uxTemplate.Text = "Template : " + collectionData.Template;
                uxFolderId.Text = "Folder ID : " + collectionData.FolderId;
                uxRecursive.Text = "Recursive : " + collectionData.Recursive;
                uxDescription.Text = "Description :" + collectionData.Description;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Collection does not exist.
", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.CollectionCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Collection Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Content Collection Property
- * Object Value

Parameters

- `criteria`. CollectionCriteria used to retrieve content collections.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentCollectionPropertyLabel"
```

```
AssociatedControlID="uxContentCollectionProperty" CssClass="span-5 last" runat="server"
Text="ContentCollectionProperty:" />
        <asp:DropDownList ID="uxContentCollectionProperty" runat="server">
            <asp:ListItem>Collection Id</asp:ListItem>
            <asp:ListItem>Collection Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-5" runat="server"
ValidationGroup="RegisterValidationGroup" Text="7" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    <asp:ListView ID="uxCollectionlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
                        <th>
                            Title
                        </th>
                        <th>
                            Folder Id
                        </th>
                        <th>
                            Last Edit Date
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("Title")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("FolderId")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("LastEditDate")%>
```

```
                </td>
            </tr>
        </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            CollectionCriteria criteria = new CollectionCriteria();
            if (uxContentCollectionProperty.SelectedItem.Text == "Collection Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(ContentCollectionProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(ContentCollectionProperty.Title,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            CollectionManager collectionManager = new CollectionManager();
            List<ContentCollectionData> collectionList = collectionManager.GetList
(criteria);

            uxCollectionlistView.Visible = true;
            uxCollectionlistView.DataSource = collectionList;
            uxCollectionlistView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Update

```
Update(Ektron.Cms.ContentCollectionData)
```

Updates an existing Content Collection item in the CMS.

Authenticated users

- CMS Administrators
- Collection Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- Title
- * Recursive
- Summary

Parameters

- `contentCollectionData`. The ContentCollectionData object to update.

Returns

Returns the custom CmsData object update.

Remarks

Validate the content collection item with `GetItem()` before you update the properties. Then, call `Update` with your modified ContentCollectionData object.

NOTE: You cannot change all properties after the initial Add event, such as the ContentCollectionData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="Label1" AssociatedControlID="uxCollectionId" CssClass="span-
3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField  ID="uxCollectionId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCollectionTitleLabel"
AssociatedControlID="uxCollectionTitle" CssClass="span-3 last" runat="server"
Text="Title:" />
```

```
        <ektronUI:TextField  ID="uxCollectionTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRecursiveLabel" AssociatedControlID="uxRecursive"
CssClass="span-3 last" runat="server" Text="* Recursive:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxRecursive" CssClass="span-
6" runat="server"  ValidationGroup="RegisterValidationGroup" />
    </li>

        <li class="clearfix">
        <ektronUI:Label ID="uxCollectionDescriptionLabel"
AssociatedControlID="uxCollectionDescription" CssClass="span-3 last" runat="server"
Text="Summary:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxCollectionDescription"
CssClass="span-4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long collectionId = long.Parse(uxCollectionId.Text);

            CollectionManager collectionManager = new CollectionManager();
            ContentCollectionData contentcollData = collectionManager.GetItem
(collectionId);


            if (contentcollData != null)
            {
                contentcollData.Title = uxCollectionTitle.Text != string.Empty ?
uxCollectionTitle.Text : contentcollData.Title;
```

```
                contentcollData.Recursive = uxRecursive.Checked;
                contentcollData.Description = uxCollectionDescription.Text !=
string.Empty ? uxCollectionDescription.Text : contentcollData.Description;
                collectionManager.Update(contentcollData);

                MessageUtilities.UpdateMessage(uxMessage, "Collection with Id " +
contentcollData.Id + " updated.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Collection does not exist",
Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

UpdateItemOrder

```
UpdateItemOrder(System.Int64,System.Collections.Generic.IEnumerable)
```

Updates the contents of a collection.

Authenticated users

- CMS Administrators
- Collection Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Collection ID
- * Content ID List (comma separated collection's content IDs)

Parameters

- `collectionId`. The ID of the content collection.
- `contentIdlist`. The IDs of the contents.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxCollectionIdLabel" AssociatedControlID="uxCollectionId"
CssClass="span-3 last" runat="server" Text="* Collection Id:" />
        <ektronUI:TextField  ID="uxCollectionId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdListLabel" AssociatedControlID="uxContentIdList"
CssClass="span-3 last" runat="server" Text="* Content Id List: (Comma Separated
collection's content Ids)" />
        <ektronUI:TextField  ID="uxContentIdList" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update Item Order"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long collectionId = long.Parse(uxCollectionId.Text);
            long contentId;

            CollectionManager collectionManager = new CollectionManager();
            ContentCollectionData collectionData = collectionManager.GetItem
(collectionId);

            //Check Whether Entered  CollectionId is Valid or Not.
            if (collectionData != null)
```

```
            {
                List<long> contentIdList = new List<long>();

                foreach (string strId in uxContentIdList.Text.Split(new char[] { ',' }))
                {
                    contentId = Convert.ToInt64(strId);

                    ContentManager contentManager = new ContentManager();
                    ContentData cData = new ContentData();
                    cData = contentManager.GetItem(contentId, false);
                    //Check Whether Entered  ContentId is Valid or Not.
                    if (cData != null)
                    {
                        contentIdList.Add(contentId);
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
ContentID.", Message.DisplayModes.Error);
                        return;
                    }
                }
                collectionManager.UpdateItemOrder(collectionId, contentIdList);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Collection does not exist.",
Message.DisplayModes.Error);
                return;
            }

            MessageUtilities.UpdateMessage(uxMessage, "Item Order Updated",
Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

CollectionCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms
```

Properties

- `CollectionCriteria()`

```
public CollectionCriteria()
```

- `CollectionCriteria(Ektron.Cms.Common.CollectionProperty, EkEnumeration.OrderByDirection)`

```
public CollectionCriteria(Ektron.Cms.Common.CollectionProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `CollectionProperty` are:

- DateCreated
- Description
- FolderId
- HasBeenPublished
- Id
- IsApprovalRequired
- IsPublished
- LastEditDate
- Status
- Template
- Title
- UserId

ContentCollectionData

Namespace

```
Ektron.Cms.Organization
```

Properties

- `DateCreated`. Gets or sets the date thiscCollection was created.

```
public System.DateTime DateCreated { set; get; }
```

- `Description`. Gets or sets the collection's description.

```
public string Description { set; get; }
```

- DynamicReplicationMethod. Gets or sets a value representing the dynamic replication method to utilize.

```
public bool DynamicReplicationMethod { set; get; }
```

- `EditorFName`. Gets ot sets the Editor's First Name.

```
public string EditorFName { set; get; }
```

- `EditorLName`. Gets or sets the Editor's Last Name.

```
public string EditorLName { set; get; }
```

- `EnableReplication`. Gets or sets a value indicating whether replication is enabled on this item (1=Enabled, 0=Disabled).

```
public int EnableReplication { set; get; }
```

- `FolderId`. Path Folder Id.

```
public long FolderId { set; get; }
```

- `Id`. Gets or sets the ID that identified this object uniquely.

```
public long Id { set; get; }
```

- `IsApprovalRequired`. Gets or sets a true or false value indicating whether approval is required for this Collection.

```
public bool IsApprovalRequired { set; get; }
```

- `LastEditDate`. Gets or sets the last Edit Date.

```
public System.DateTime LastEditDate { set; get; }
```

- `Published`. Gets or sets a integer value that indicates whether the Collection is published or not (1=Published, 0=Not Published).

```
public bool Published { set; get; }
```

- `Recursive`. Gets or sets a true or false value indicating whether to include subfolders.

```
public bool Recursive { set; get; }
```

- `Status`. Content Status:
  - A-Approved
  - D-Pending deletion
  - I-Checked in
  - M-Marked for deletion
  - O-Checked out
  - P-Pending go live date
  - S-Submitted for approval
  - T-Awaiting completion of associated tasks

```
public string Status { set; get; }
```

- `Template`. Gets or sets a template URL for the items in the collection, leave blank to use the item's quicklinks.

```
public string Template { set; get; }
```

- `Title`. Gets or sets the collection's title.

```
public string Title { set; get; }
```

- `UserId`. Gets or sets the user id that represents the author of this collection.

```
public long UserId { set; get; }
```

- `Validate()`. Validate this collection. Returns true for valid data, false for invalid data.

```
public override ValidationResults Validate()
```

FolderManager

8.50 and higher

The FolderManager class manages Ektron CMS folders. See also Setting Up Your CMS Folder Structure in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Organization.FolderManager
```

Constructors

- `FolderManager()`
- `FolderManager(ApiAccessMode)`

Properties

- `FolderService`. Gets instance to IFolder.

Methods

- *Add* on the next page
- *AllowTaxonomy* on page 1234
- *AssignMetadata* on page 1236
- `CopyFolder(Int64,Int64,Boolean,Boolean)`. 8.60 and higher Copy the folder from one parent folder to another parent folder.
- *Delete* on page 1238
- *DisableTaxonomy* on page 1239
- *EnableTaxonomy* on page 1241
- `GetAssignedFolders(Int64)`. 8.60 and higher Retrieves the list of folders assigned to the given taxonomy.
- `GetAssignedMetadata(Int64)`. 8.60 and higher Returns a list of content metadata assigned to a given folder.
- *GetAssignedTaxonomy* on page 1243
- *GetItem* on page 1246
- *GetList* on page 1248
- `GetSitemapPath(Int64)`. 8.60 and higher Returns the specified `sitemappath` object for a given folder. To get sitemappaths in different languages, change the manager's `ekRequestInformation` Language.
- `GetTree(Int64)`. 8.60 and higher

- `IsCommunityFolder(Int64)`. 8.60 and higher Returns true if the folder is a Community Folder.
- *LinkTaxonomy* on page 1251
- `MoveFolder(Int64,Int64,Boolean)`. 8.60 and higher Move the folder from existing parent folder to another parent folder.
- `Purge(Int64,DateTime,Boolean,Boolean)`. 8.60 and higher Purge the content history values for the folder until the specified date criteria.
- *UnallowTaxonomy* on page 1253
- *UnassignMetadata* on page 1255
- `UnAssignTemplate(Int64,Int64)`. 9.00 and higher UnAssign the template from the folder.
- `UnlinkTaxonomy(Int64,Int64)`. 8.60 and higher Unlink the supplied folder from the supplied taxonomy. All content in the folder is automatically unassigned from the given taxonomy.
- *Update* on page 1257

Add

```
Add(Ektron.Cms.FolderData)
```

Adds a folder based on information in an FolderData object.

The method populates the `message.Id` with the newly created ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Name
- * Parent Folder ID
- Style Sheet
- Description

Parameters

- `folderData`. The FolderData object to add.

Returns

Added FolderData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-3
last" runat="server" Text="* Name:" />
        <ektronUI:TextField ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxParentFolderIdLabel"
AssociatedControlID="uxParentFolderId" CssClass="span-3 last" runat="server" Text="*
Parent Folder Id:" />
        <ektronUI:TextField ID="uxParentFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStyleSheetLabel" AssociatedControlID="uxStyleSheet"
CssClass="span-3 last" runat="server" Text=" Style Sheet:" />
        <ektronUI:TextField ID="uxStyleSheet" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-3 last" runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxDescription" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
```

```
            FolderManager folderManager = new FolderManager();
            FolderData folderData = new FolderData()
            {
                 Name = uxName.Text,
                 ParentId = long.Parse(uxParentFolderId.Text),
                 Description = uxDescription.Text,
                 StyleSheet = uxStyleSheet.Text
            };

            folderManager.Add(folderData);

            MessageUtilities.UpdateMessage(uxMessage, "Folder Added with Id " +
folderData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

AllowTaxonomy

```
AllowTaxonomy(System.Int64,System.Collections.Generic.List)
```

Allows a list of taxonomy data based on IDs to a folder.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID
- * Taxonomy ID List (comma separated)

Parameters

- `folderId`. Folder ID to which taxonomies are permitted.
- `taxonomyIdList`. List of taxonomy IDs.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last"
            runat="server" Text="* Folder Id:" />
```

```
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0"   />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIdListLabel"
AssociatedControlID="uxTaxonomyIdList" CssClass="span-3 last"
            runat="server" Text="* Taxonomy Id List: (Comma Separated TaxonomyIds)" />
        <ektronUI:TextField ID="uxTaxonomyIdList" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Assign Taxonomy"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3 last" runat="server"
Text="* - Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Framework.Content;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long folderId = long.Parse(uxFolderId.Text);
            long taxonomyId;
            List<long> taxonomyIdList = new List<long>();

            FolderManager folderManager = new FolderManager();
            FolderData folderData = folderManager.GetItem(folderId);
            if (folderData != null)
            {
                foreach (string strId in uxTaxonomyIdList.Text.Split(new char[] { ','
}))
                {
                    taxonomyId = Convert.ToInt64(strId);
```

```
                    TaxonomyManager taxonomyManager = new TaxonomyManager();
                    TaxonomyData taxonomyData = taxonomyManager.GetItem(taxonomyId);
                    //Check Whether Entered  TaxonomyId is Valid or Not.
                    if (taxonomyData != null)
                    {
                        taxonomyIdList.Add(taxonomyId);
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        uxStatus.Text = "Please Enter Valid TaxonomyId.";
                        return;
                    }
                }
                folderManager.AllowTaxonomy(folderId, taxonomyIdList);
                uxMessage.Text = "Taxonomy ID's assigned to folder with Id  " +
folderId;
            }
            else
            {
                uxStatus.Visible = true;
                uxStatus.Text = "Please Enter Valid FolderID.";
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            uxMessage.Text = ex.Message;
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

AssignMetadata

8.60 and higher

```
AssignMetadata(System.Int64,System.Int64,System.Boolean)
```

Assign a metadata to a folder.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID
- * Metadata Type ID

- Is Required

Parameters

- `folderId`. The ID of the folder.
- `metadataTypeId`. The ID of the metadata type.
- `isRequired`. Indicates whether metadata is required.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last" runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMetadataIdLabel" AssociatedControlID="uxMetadataId"
CssClass="span-3 last" runat="server" Text="* MetadataType Id:" />
        <ektronUI:TextField ID="uxMetadataId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxIsRequiredLabel" AssociatedControlID="uxIsRequired"
CssClass="span-3 last" runat="server" Text=" Is Required:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsRequired"
CssClass="span-6" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Assign Metadata"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3 last" runat="server"
Text="* - Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
```

```
        {
            long folderId = long.Parse(uxFolderId.Text);
            long metadataTypeId = long.Parse(uxMetadataId.Text);

            FolderManager folderManager = new FolderManager();
            folderManager.AssignMetadata(folderId, metadataTypeId,uxIsRequired.Checked);

            MessageUtilities.UpdateMessage(uxMessage,  "Metadata Type assigned to the
folder", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a folder from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `folderId`. The ID of the folder to delete.

Remarks

Validate the item using `GetItem()` before deleting.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
```

```
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long folderID = long.Parse(uxFolderId.Text);
            FolderManager folderManager = new FolderManager();
            FolderData folderData = folderManager.GetItem(folderID);
            if (folderData != null)
            {
                folderManager.Delete(folderID);
                MessageUtilities.UpdateMessage(uxMessage, "Folder with Id " +
uxFolderId.Text + " deleted", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Folder with Id " +
uxFolderId.Text + " is Invalid", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DisableTaxonomy

8.60 and higher

```
DisableTaxonomy(System.Int64,System.Int64)
```

Disables the supplied taxonomy to be used on content in the folder.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID
- * Taxonomy ID

Parameters

- `folderId`. Folder ID to which taxonomies are permitted.
- `taxonomyId`. Taxonomy ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="Label1" AssociatedControlID="uxFolderId" CssClass="span-3
last"
            runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="TextField1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="Label2" AssociatedControlID="uxTaxonomyId"
            CssClass="span-3 last" runat="server" Text="* Taxonomy Id" />
        <ektronUI:TextField ID="TextField2" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="Label3" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="Button1" runat="server" OnClick="uxSubmit_Click"
Text="Assign Taxonomy"></ektronUI:Button>
        <ektronUI:Label ID="Label4" CssClass="span-3 last" runat="server" Text="* -
Required" />
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Framework.Content;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long folderId = long.Parse(uxFolderId.Text);
            long taxonomyId;

            FolderManager folderManager = new FolderManager();
            FolderData folderData = folderManager.GetItem(folderId);
            if (folderData != null)
            {
                                taxonomyId = Convert.ToInt64(uxTaxonomyId.Text);
                                folderManager.DisableTaxonomy(folderId, taxonomyId);
                uxMessage.Text = "Taxonomy ID's is not longer allowed with the folder Id
 " + folderId;
            }
            else
            {
                uxStatus.Visible = true;
                uxStatus.Text = "Please Enter Valid FolderID.";
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            uxMessage.Text = ex.Message;
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

EnableTaxonomy

8.60 and higher

```
EnableTaxonomy(System.Int64,System.Collections.Generic.List)
```

Enables these taxonomies to be assigned to content items in the supplied folder.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID
- * Taxonomy ID List (comma separated)

Parameters

- `folderId`. Folder ID to which taxonomies are permitted.
- `taxonomyIdList`. List of taxonomy IDs.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last"
            runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0"   />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIdListLabel"
AssociatedControlID="uxTaxonomyIdList" CssClass="span-3 last"
            runat="server" Text="* Taxonomy Id List: (Comma Separated TaxonomyIds)" />
        <ektronUI:TextField ID="uxTaxonomyIdList" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Assign Taxonomy"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3 last" runat="server"
Text="* - Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Framework.Content;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long folderId = long.Parse(uxFolderId.Text);
            long taxonomyId;
            List<long> taxonomyIdList = new List<long>();

            FolderManager folderManager = new FolderManager();
            FolderData folderData = folderManager.GetItem(folderId);
            if (folderData != null)
            {
                foreach (string strId in uxTaxonomyIdList.Text.Split(new char[] { ','
}))
                {
                    taxonomyId = Convert.ToInt64(strId);

                    TaxonomyManager taxonomyManager = new TaxonomyManager();
                    TaxonomyData taxonomyData = taxonomyManager.GetItem(taxonomyId);
                    //Check Whether Entered  TaxonomyId is Valid or Not.
                    if (taxonomyData != null)
                    {
                        taxonomyIdList.Add(taxonomyId);
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        uxStatus.Text = "Please Enter Valid TaxonomyId.";
                        return;
                    }
                }
                folderManager.EnableTaxonomy(folderId, taxonomyIdList);
                uxMessage.Text = "Taxonomy ID's assigned to folder with Id  " +
folderId;
            }
            else
            {
                uxStatus.Visible = true;
                uxStatus.Text = "Please Enter Valid FolderID.";
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            uxMessage.Text = ex.Message;
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetAssignedTaxonomy

```
GetAssignedTaxonomy(System.Int64)
```

Retrieves a list of Taxonomy data that is assigned to a folder.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID

Parameters

- `folderId`. Folder ID for which to retrieve taxonomy data.

Returns

List of TaxonomyData.

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last"
                runat="server" Text="* Folder Id:" />
            <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
        </li>
        <li class="clearfix" style="color: red;">
            <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                Text="Status:" />
        </li>
        <li class="clearfix">
            <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetAssigned Taxonomy"></ektronUI:Button>
            <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3 last" runat="server"
Text="* - Required" />
        </li>
    </ol>

    <ol class="formFields">
        <li class="clearfix">
            <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
        </li>
        <asp:ListView ID="uxTaxonomylistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
            <EmptyDataTemplate >There are No Taxonomy's assigned to
Folder.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
```

```
                <thead>
                    <tr>
                        <th>
                            TaxonomyId
                        </th>
                        <th>
                            TaxonomyName
                        </th>
                        <th>
                            TaxonomyPath
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("TaxonomyId")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("TaxonomyName")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("TaxonomyPath")%>
                </td>
            </tr>
        </ItemTemplate>
    </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Framework.Content;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long folderId = long.Parse(uxFolderId.Text);

            FolderManager folderManager = new FolderManager();
```

```
            FolderData folderData = folderManager.GetItem(folderId);
            if (folderData != null)
            {
                List<TaxonomyBaseData> taxonomyIds = folderManager.GetAssignedTaxonomy
(folderId);
                MessageUtilities.UpdateMessage(uxMessage,  "Taxonomy's assigned to
folder with Id  " + folderId + " are below.", Message.DisplayModes.Success);

                uxTaxonomylistView.Visible = true;
                uxTaxonomylistView.DataSource = taxonomyIds;
                uxTaxonomylistView.DataBind();
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid FolderID.",
Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.FolderData)
```

Retrieves the properties of a specific FolderData object.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Recursive

Parameters

- `folderId`. The ID of the folder to retrieve.
- `returnChildProperties`. Indicates whether child properties are returned.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
            <ektronUI:Label ID="uxRecursiveLabel" AssociatedControlID="uxRecursive"
CssClass="span-3 last"
                runat="server" Text="* Recursive :" />
            <asp:DropDownList ID="uxRecursive" runat="server">
                <asp:ListItem>false</asp:ListItem>
                <asp:ListItem>true</asp:ListItem>
            </asp:DropDownList>
    </li>
    <li class="clearfix" style="color: red;">
            <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                Text="Status:" />
        </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTemplate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTotalContent" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxPrivateContent" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long folderID = long.Parse(uxFolderId.Text);
            bool recursive = Convert.ToBoolean(uxRecursive.SelectedItem.ToString());
            FolderManager folderManager = new FolderManager();

            FolderData folderData = folderManager.GetItem(folderID, recursive);

            if (folderData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for folder with ID "
+ folderData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxName.Text = "Name : " + folderData.Name ;
                uxTemplate.Text = "Template : " + folderData.TemplateFileName;
                uxDescription.Text = "Description :" + folderData.Description;
                uxPrivateContent.Text = "Private Content :" +
folderData.PrivateContent.ToString();
                uxTotalContent.Text = "Total Content :" +
folderData.TotalContent.ToString();
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Folder does not exist. ",
Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.FolderCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Folder Property
- * Object Value

Parameters

- `criteria`. FolderCriteria used to retrieve folders.

Returns

List of retrieved FolderData items.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderPropertyLabel"
AssociatedControlID="uxFolderProperty" CssClass="span-4 last" runat="server"
Text="Folder Property:" />
        <asp:DropDownList ID="uxFolderProperty" runat="server">
            <asp:ListItem>Folder Id</asp:ListItem>
            <asp:ListItem>Folder Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last" runat="server" Text="* Object Value:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxFolderlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
```

```
            <thead>
                <tr>
                    <th>
                        Folder Name
                    </th>
                    <th>
                        Folder Id
                    </th>
                    <th>
                        Folder Path
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("NameWithPath")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Organization;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            FolderCriteria criteria = new FolderCriteria();
```

```
            if (uxFolderProperty.SelectedItem.Text == "Folder Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(FolderProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(FolderProperty.FolderName ,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            FolderManager folderManager = new FolderManager();
            List<FolderData> folderList = folderManager.GetList(criteria);

            uxFolderlistView.Visible = true;
            uxFolderlistView.DataSource = folderList;
            uxFolderlistView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

LinkTaxonomy

```
LinkTaxonomy(System.Int64,System.Int64)
```

Links the taxonomy to a folder.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID
- * Taxonomy ID

Parameters

- `criteria`. FolderCriteria used to retrieve folders.

Returns

List of retrieved FolderData items.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last"
            runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0"   />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIdLabel" AssociatedControlID="uxTaxonomyId"
CssClass="span-3 last"
            runat="server" Text="* Taxonomy Id" />
        <ektronUI:TextField ID="uxTaxonomyId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Assign Taxonomy"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3 last" runat="server"
Text="* - Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Framework.Content;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
            try
        {
            var folderId = long.Parse(uxFolderId.Text);
                    var taxonomyId = long.Parse(uxTaxonomyId.Text);
```

```
            var folderManager = new FolderManager
                                    {
                                            ContentLanguage = 1033
                                    };
               var folderData = folderManager.GetItem(folderId);
                       var taxonomyManager = new TaxonomyManager();
                       var taxonomyData = taxonomyManager.GetItem(taxonomyId);

               if (folderData == null)
            {
                                uxStatus.Visible = true;
                                uxStatus.Text = "Please Enter Valid FolderID.";
                                return;
            }

                       if (taxonomyData == null)
                       {
                                uxStatus.Visible = true;
                                uxStatus.Text = "Please Enter Valid TaxonomyID.";
                                return;
                       }

                       folderManager.LinkTaxonomy(folderId, taxonomyId);

            uxPageMultiView.SetActiveView(uxViewMessage);
               uxMessage.Text = String.Format("Folder with Id {0} has been linked to taxonomy
with id {1}", folderId,
                                                taxonomyId);
        }
        catch (Exception ex)
        {
            uxMessage.Text = ex.Message;
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

UnallowTaxonomy

```
UnallowTaxonomy(System.Int64,System.Collections.Generic.List)
```

Disallows a taxonomy for a folder.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID
- * Taxonomy ID List

Parameters

- `folderId`. The folder ID.
- `taxonomyId`. The taxonomy ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="Label1" AssociatedControlID="uxFolderId" CssClass="span-3
last"
            runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="TextField1" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="Label2" AssociatedControlID="uxTaxonomyId"
            CssClass="span-3 last" runat="server" Text="* Taxonomy Id" />
        <ektronUI:TextField ID="TextField2" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="Label3" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="Button1" runat="server" OnClick="uxSubmit_Click"
Text="Assign Taxonomy"></ektronUI:Button>
        <ektronUI:Label ID="Label4" CssClass="span-3 last" runat="server" Text="* -
Required" />
    </li>
</ol>


<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Framework.Content;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long folderId = long.Parse(uxFolderId.Text);
```

```
            long taxonomyId;

            FolderManager folderManager = new FolderManager();
            FolderData folderData = folderManager.GetItem(folderId);
            if (folderData != null)
            {
                                taxonomyId = Convert.ToInt64(uxTaxonomyId.Text);
                                folderManager.UnallowTaxonomy(folderId, taxonomyId);
                uxMessage.Text = "Taxonomy ID's is not longer allowed with the folder Id
" + folderId;
            }
            else
            {
                uxStatus.Visible = true;
                uxStatus.Text = "Please Enter Valid FolderID.";
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            uxMessage.Text = ex.Message;
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

UnassignMetadata

```
UnassignMetadata(System.Int64,System.Int64)
```

Unassign a metadata from a folder.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID
- * Metadata Type ID

Parameters

- `folderId`. The ID of the folder.
- `metadataTypeId`. The ID of the metadata type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last" runat="server" Text="* Folder Id:" />
        <ektronUI:TextField ID="uxFolderId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMetadataIdLabel" AssociatedControlID="uxMetadataId"
CssClass="span-3 last" runat="server" Text="* MetadataType Id:" />
        <ektronUI:TextField ID="uxMetadataId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Unassign Metadata"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3 last" runat="server"
Text="* - Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            long folderId = long.Parse(uxFolderId.Text);
            long metadataTypeId = long.Parse(uxMetadataId.Text);

            FolderManager folderManager = new FolderManager();
            FolderData folderData = folderManager.GetItem(folderId);

            if (folderData != null)
```

```
            {
                MetadataTypeManager metadataTypeManager = new MetadataTypeManager();
                ContentMetaData metaData = metadataTypeManager.GetItem(metadataTypeId);

                if (metaData != null)
                {
                    folderManager.UnassignMetadata(folderId, metadataTypeId);
                    MessageUtilities.UpdateMessage(uxMessage, "Metadata Type unassigned
from the folder", Message.DisplayModes.Success);
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Metadata type does not
exist. ", Message.DisplayModes.Error);
                    return;
                }

            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Folder does not exist. ",
Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.MenuBaseData)
```

Updates an existing folder in the CMS.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Folder ID
- Name

- Publish as PDF
- Style Sheet
- Description

Parameters

- `folderData`. The FolderData object to update.

Returns

Updated FolderData object.

Remarks

Validate the folder with `GetItem()` before you update the properties. Then, call `Update` with your modified FolderData object.

NOTE: You cannot change all properties after the initial Add event, such as the FolderData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFolderIdLabel" AssociatedControlID="uxFolderId"
CssClass="span-3 last" runat="server" Text="* Folder Id:" />
        <ektronUI:TextField  ID="uxFolderId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-3
last" runat="server" Text="  Name:" />
        <ektronUI:TextField  ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxPublishPdfLabel" AssociatedControlID="uxPublishPdf"
CssClass="span-3 last" runat="server" Text=" Publish As PDF:" />
        <ektronUI:Button  ID="uxPublishPdf" DisplayMode="Checkbox" Text=""
CssClass="span-4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStyleSheetLabel" AssociatedControlID="uxStyleSheet"
CssClass="span-3 last" runat="server" Text=" Style Sheet:" />
        <ektronUI:TextField ID="uxStyleSheet" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-3 last" runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxDescription" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
```

```
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long folderID = long.Parse(uxFolderId.Text);
            FolderManager folderManager = new FolderManager();
            FolderData folderData = folderManager.GetItem(folderID);
            if (folderData != null)
            {
                folderData.Name = (uxName.Text != string.Empty ? uxName.Text :
folderData.Name);
                folderData.StyleSheet = (uxStyleSheet.Text != string.Empty ?
uxStyleSheet.Text : folderData.StyleSheet);
                folderData.Description = (uxDescription.Text != string.Empty ?
uxDescription.Text : folderData.Description);
                folderData.PublishPdfActive = uxPublishPdf.Checked;
                folderData.PublishPdfEnabled = uxPublishPdf.Checked;
                folderManager.Update(folderData);
                MessageUtilities.UpdateMessage(uxMessage,  "Folder with Id " +
folderData.Id + " updated", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Folder does not exists.",
Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Data Classes

FolderCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms
```

Constructors

- `FolderCriteria()`

  ```
  public FolderCriteria()
  ```

- `FolderCriteria(Ektron.Cms.Common.FolderProperty, EkEnumeration.OrderByDirection)`

  ```
  public FolderCriteria(Ektron.Cms.Common.FolderProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `FolderProperty` are:

  - Description
  - FolderName
  - FolderPath
  - FolderType
  - Id
  - IsCommunityFolder
  - IsDomainFolder
  - IsHidden
  - IsPrivate
  - ParentId

Properties

- `ReturnChildProperties`

  ```
  public bool ReturnChildProperties { set; get; }
  ```

FolderData

Namespace

```
Ektron.Cms
```

Properties

- `AliasRequired`

  ```
  public bool AliasRequired { set; get; }
  ```

- ApprovalMethod

```
public int ApprovalMethod { set; get; }
```

- CategoryRequired

```
public bool CategoryRequired { set; get; }
```

- ChildFolders

```
public Ektron.Cms.Controls.CmsWebService.FolderData[] ChildFolders { set; get; }
```

- Description

```
public string Description { set; get; }
```

- DomainProduction

```
public string DomainProduction { set; get; }
```

- DomainStaging

```
public string DomainStaging { set; get; }
```

- FlagInheritedFrom

```
public long FlagInheritedFrom { set; get; }
```

- FolderFlags

```
public Ektron.Cms.Controls.CmsWebService.FlagDefData[] FolderFlags { set; get; }
```

- FolderMetadata

```
public Ektron.Cms.Controls.CmsWebService.ContentMetaData[] FolderMetadata { set;
get; }
```

- FolderTemplates

```
public Ektron.Cms.Controls.CmsWebService.TemplateData[] FolderTemplates { set; get;
}
```

- FolderType

```
public int FolderType { set; get; }
```

- HasChildren

```
public bool HasChildren { set; get; }
```

- IsAliasInherited

```
public bool IsAliasInherited { set; get; }
```

- IsCommunityFolder

```
public bool IsCommunityFolder { set; get; }
```

- IsDomainFolder

```
public bool IsDomainFolder { set; get; }
```

- IsFlagInherited

```
public bool IsFlagInherited { set; get; }
```

- IsMetaInherited

```
public bool IsMetaInherited { set; get; }
```

- IsPermissionInherited

```
public bool IsPermissionsInherited { set; get; }
```

- IsSitemapInherited

```
public bool IsSitemapInherited { set; get; }
```

- IsStyleSheetInherited

```
public bool IsStyleSheetInherited { set; get; }
```

- IsTaxonomyInherited

```
public bool IsTaxonomyInherited { set; get; }
```

- IsTemplateInherited

```
public bool IsTemplateInherited { set; get; }
```

- IsXmlInherited

```
public bool IsXmlInherited { set; get; }
```

- Name

```
public string Name { set; get; }
```

- NameWithPath

```
public string NameWithPath { set; get; }
```

- ParentId

```
public long ParentId { set; get; }
```

- Permissions

```
public Ektron.Cms.Controls.CmsWebService.PermissionData Permissions { set; get; }
```

- PrivateContent

```
public bool PrivateContent { set; get; }
```

- PublishHtmlActive

```
public bool PublishHtmlActive { set; get; }
```

- PublishPdfActive

```
public bool PublishPdfActive { set; get; }
```

- PublishPdfEnabled

```
public bool PublishPdfEnabled { set; get; }
```

- ReplicationMethod

```
public int ReplicationMethod { set; get; }
```

- SitemapPath

```
public Ektron.Cms.Controls.CmsWebService.SitemapPath[] SitemapPath { set; get; }
```

- StyleSheet

```
public string StyleSheet { set; get; }
```

- TemplateFileName

```
public string TemplateFileName { set; get; }
```

- TemplateId

```
public long TemplateId { set; get; }
```

- TotalContent

```
public int TotalContent { set; get; }
```

- XmlConfiguration

```
public Ektron.Cms.Controls.CmsWebService.XmlConfigData[] XmlConfiguration { set; get; }
```

MenuManager

8.50 and higher

The MenuManager class manages drop down menus for your Web site.

Menu items can link to content, library assets, external hyperlinks, and submenus.

Namespace

```
Ektron.Cms.Framework.Organization.MenuManager
```

Constructors

- `MenuManager()`
- `MenuManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `MenuManagerService`. Returns an instance of the business objects task manager service.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- `GetMenuItemList(MenuItemCriteria)`. 9.00 and higher Returns a list of menu ites based upon the supplied MenuItemCriteria.
- `GetMenuList(MenuCriteria)`. 8.60 and higher Gets a list of MenuData, not menubase data, based upon the supplied MenuItemCriteria.
- *GetTree* on page 1278
- *Update (menu)* on page 1281
- *Update (menu item)* on page 1284
- *Update (submenu)* on page 1287

Add (menu)

```
Add(Ektron.Cms.MenuData)
```

Adds the specified MenuData object.

Authenticated users

- CMS Administrators
- Menu Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- Description

Parameters

- `menuData`. The MenuData.

Remarks

Validate the ID using `GetItem()` before adding a menu for Library or Content types item.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-3
last" runat="server" Text="* Title:" />
        <ektronUI:TextField ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-3 last" runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxDescription" CssClass="span-4"
```

```
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MenuManager menuManager = new MenuManager();
            MenuData menuData = new MenuData();

            menuData.Text = uxName.Text;
            menuData.Description = uxDescription.Text;

            menuManager.Add(menuData);

            MessageUtilities.UpdateMessage(uxMessage, "Menu Added with Id " +
menuData.Id, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Add (menu item)

```
Add(Ektron.Cms.MenuItemData)
```

Adds the specified MenuItemData.

Authenticated users

- CMS Administrators
- Menu Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- * Menu or Submenu ID
- Description

Parameters

- `menuItemData`. The MenuItemData.

Remarks

Validate the ID using `GetItem()` before adding a menu for Library or Content types item.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxItemIdLabel" AssociatedControlID="uxItemId"
CssClass="span-4 last"
            runat="server" Text="*Item Id (ContentId):" />
        <ektronUI:TextField ID="uxItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxMenuIdLabel" AssociatedControlID="uxMenuId"
CssClass="span-4 last"
            runat="server" Text="* Id (Menu/Submenu):" />
        <ektronUI:TextField ID="uxMenuId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>

    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last"
            runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3" ID="uxDescription" CssClass="span-4"
runat="server"
            ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Message ID="uxStatus" DisplayMode="Error" runat="server"
Visible="false" />
    </li>
        <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MenuManager menuManager = new MenuManager();
            ContentMenuItemData menuItemData = new ContentMenuItemData();

            long itemId = long.Parse(uxItemId.Text);
            long menuId = long.Parse(uxMenuId.Text);

            MenuData menuData = menuManager.GetMenu(menuId);
            if (menuData == null)
            {
               uxStatus.Visible = true;
               MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Menu ID.",
Message.DisplayModes.Error);
               return;
            }

               ContentManager contentManager = new ContentManager();
            ContentData cData = contentManager.GetItem(itemId);
```

```
            if (cData != null)
            {
                menuItemData.ItemId = itemId;
                menuItemData.Description = uxDescription.Text;
                menuItemData.Text = cData.Title;
                menuItemData.ParentId = menuData.Id;

                menuManager.Add(menuItemData);

                MessageUtilities.UpdateMessage(uxMessage, "MenuItem Added with Id " +
menuItemData.Id, Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Content
ID.", Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Add (submenu)

```
Add(Ektron.Cms.SubMenuData)
```

Adds a submenu to the CMS from the SubMenuData object.

Authenticated users

- CMS Administrators
- Menu Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- * Submenu ID
- Description

Parameters

- `subMenuData`. The SubMenuData.

Remarks

Validate the ID using `GetItem()` before adding a menu for Library or Content types item.

.aspx code snippet

```
<ol class="formFields">
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxMenuIdLabel" AssociatedControlID="uxMenuId"
CssClass="span-4 last"
            runat="server" Text="* Id (Menu/Submenu):" />
        <ektronUI:TextField ID="uxMenuId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxMenuTitleLabel" AssociatedControlID="uxMenuTitle"
CssClass="span-4 last"
            runat="server" Text="*Title :" />
            <asp:TextBox  Rows="3" ID="uxMenuTitle" CssClass="span-4" runat="server"
            ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last"
            runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3" ID="uxDescription" CssClass="span-4"
runat="server"
            ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Message ID="uxStatus" DisplayMode="Error" runat="server"
Visible="false" />
    </li>
        <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MenuManager menuManager = new MenuManager();

            long menuId = long.Parse(uxMenuId.Text);

            MenuData menuData = menuManager.GetMenu(menuId);
            if (menuData == null)
            {
               uxStatus.Visible = true;
               MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Menu ID.",
Message.DisplayModes.Error);
               return;
            }

               SubMenuData submenuData = new SubMenuData();
            submenuData.Text = uxMenuTitle.Text;
            submenuData.Description = uxDescription.Text;
                       submenuData.ParentId = menuData.Id;

            menuManager.Add(submenuData);

            MessageUtilities.UpdateMessage(uxMessage, "Sub Menu Added with Id " +
submenuData.Id, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeleteMenu

```
DeleteMenu(System.Int64)
```

Deletes a menu from the CMS.

Authenticated users

- CMS Administrators
- Menu Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The ID.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMenuIdLabel" AssociatedControlID="uxMenuId"
CssClass="span-3 last"
        runat="server" Text="* Menu Id:" />
        <ektronUI:TextField ID="uxMenuId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
        Text="0" />
    </li>
        <li class="clearfix" style="color: red;">
    <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last" runat="server"
        Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MenuManager menuManager = new MenuManager();
            MenuData menuBaseData = new MenuData();
            long menuId = Convert.ToInt64(uxMenuId.Text);

            menuBaseData = menuManager.GetMenu(menuId);
            if (menuBaseData != null)
            {
                menuManager.DeleteMenu(menuId);
                MessageUtilities.UpdateMessage(uxMessage,  "Menu Id " + menuId + " has
been Deleted.", Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus,  "Please Enter Valid Menu ID.",
Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeleteMenuItem

```
DeleteMenuItem(System.Int64)
```

Deletes the menu item and all of its children.

Authenticated users

- CMS Administrators
- Menu Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The ID.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
            <ektronUI:Label ID="uxItemIdLabel" AssociatedControlID="uxItemId"
CssClass="span-3 last"
            runat="server" Text="*Item Id:" />
        <ektronUI:TextField ID="uxItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
        <li class="clearfix" style="color: red;">
    <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last" runat="server"
        Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="DeleteItem"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MenuManager menuManager = new MenuManager();
            MenuItemData menuItemData = new MenuItemData();
            long itemId = Convert.ToInt64(uxItemId.Text);
```

```
            menuItemData = menuManager.GetMenuItem(itemId);
            if (menuItemData != null)
            {
                menuManager.DeleteMenuItem(itemId);
                MessageUtilities.UpdateMessage(uxMessage, "Menu Id " + itemId + " has
been Deleted.", Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus,"Please Enter Valid Menu
ItemID.", Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetMenu

```
GetMenu(System.Int64)
```

Retrieves the menu from its ID.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Menu ID

Parameters

- `id`. The ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMenuIdLabel" AssociatedControlID="uxMenuId"
CssClass="span-3 last"
            runat="server" Text="* Menu Id:" />
        <ektronUI:TextField ID="uxMenuId" CssClass="span-6" runat="server"
```

```
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxParentMenuId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxType" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            MenuManager menuManager = new MenuManager();
            MenuData menuBaseData = new MenuData();
            long menuId = Convert.ToInt64(uxMenuId.Text);
```

```
            menuBaseData = menuManager.GetMenu(menuId);
            if (menuBaseData != null)
            {

                MessageUtilities.UpdateMessage(uxMessage, "Details for Menu with ID " +
menuBaseData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxName.Text = "Name : " + menuBaseData.Text;
                uxDescription.Text = "Description : " + menuBaseData.Description;
                uxParentMenuId.Text = "ParentMenuId :" + menuBaseData.ParentId;
                uxType.Text = "Type :" + menuBaseData.Type.ToString();
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Menu ID.",
Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetMenuItem

```
GetMenuItem(System.Int64)
```

Retrieves the properties of a specific MenuData object.

Authenticated users

- CMS Administrators
- Menu Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Item ID

Parameters

- `id`. The ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxItemIdLabel" AssociatedControlID="uxItemId"
CssClass="span-3 last"
            runat="server" Text="*Item Id:" />
        <ektronUI:TextField ID="uxItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxParentMenuId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxType" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
        {
            MenuManager menuManager = new MenuManager();
            MenuItemData menuItemData = new MenuItemData();
            long itemId = Convert.ToInt64(uxItemId.Text);

            menuItemData = menuManager.GetMenuItem(itemId);

            if (menuItemData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for MenuItem with ID
" + menuItemData.Id + " are below", Message.DisplayModes.Success);

                uxName.Text = "Name : " + menuItemData.Text;
                uxDescription.Text = "Description : " + menuItemData.Description;
               uxParentId.Text = "ParentId :" + menuItemData.ParentId;
                uxType.Text = "Type :" + menuItemData.Type.ToString();
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Menu
ItemID.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTree

```
GetTree(System.Int64)
```

Retrieves the menu and its submenus.

Authenticated users

- CMS Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Menu ID

Parameters

- `Menu id`. The menu ID.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMenuIdLabel" AssociatedControlID="uxMenuId"
CssClass="span-3 last"
            runat="server" Text="* Menu Id:" />
        <ektronUI:TextField ID="uxMenuId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetTree"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxParentMenuId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTotalItems" runat="server"></asp:Literal>
    </li>
    <asp:ListView ID="uxMenulistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
```

```
                        <th>
                                Name
                        </th>
                        <th>
                                Id
                        </th>
                        <th>
                            Description
                        </th>
                        <th>
                            ParentId
                        </th>
                        <th>
                            Type
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("Text")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Id")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Description")%>
                </td>
                    <td class="devsite-method">
                    <%# Eval("ParentId")%>
                </td>
                    <td class="devsite-method">
                    <%# Eval("Type")%>
                </td>
            </tr>
        </ItemTemplate>
    </asp:ListView>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MenuManager menuManager = new MenuManager();
            long menuId = Convert.ToInt64(uxMenuId.Text);

            IMenuData menuData =  menuManager.GetTree(menuId);
            if (menuData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for Menu with ID " +
menuData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxName.Text = "Name : " + menuData.Text;
                uxDescription.Text = "Description : " + menuData.Description;
                uxParentMenuId.Text = "ParentMenuId :" + menuData.ParentId;
                uxType.Text = "Type :" + menuData.Type.ToString();
                uxTotalItems.Text = " Total Items : " + menuData.Items.Count;

                if (menuData.Items.Count > 0)
                {
                    ICollection<IMenuItemData> itemslist = menuData.Items;
                    uxMenulistView.Visible = true;
                    uxMenulistView.DataSource = itemslist;
                    uxMenulistView.DataBind();
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Menu ID.",
Message.DisplayModes.Error);
                return;
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update (menu)

```
Update(Ektron.Cms.Organization.MenuData)
```

Updates the specified MenuData in the CMS.

Authenticated users

- CMS Administrators
- Menu Administrators

- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Menu ID
- Title
- Description

Parameters

- `menuData`. The MenuData.

Remarks

Validate the menu item with `GetItem()` before you update the properties. Then, call `Update` with your modified MenuData object.

NOTE: You cannot change all properties after the initial Add event, such as the FolderData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMenuIdLabel" AssociatedControlID="uxMenuId"
CssClass="span-3 last"
        runat="server" Text="* Menu Id:" />
        <ektronUI:TextField ID="uxMenuId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
        Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-3
last" runat="server" Text=" Title:" />
        <ektronUI:TextField ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-3 last" runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxDescription" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix" style="color: red;">
    <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last" runat="server"
        Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
```

```
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMenuIdLabel" AssociatedControlID="uxMenuId"
CssClass="span-3 last"
        runat="server" Text="* Menu Id:" />
        <ektronUI:TextField ID="uxMenuId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
        Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-3
last" runat="server" Text=" Title:" />
        <ektronUI:TextField ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-3 last" runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxDescription" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix" style="color: red;">
    <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last" runat="server"
        Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MenuManager menuManager = new MenuManager();
            MenuData menuData = new MenuData();
            long menuId = Convert.ToInt64(uxMenuId.Text);

            menuData = menuManager.GetMenu(menuId);
            if (menuData != null)
            {
```

```
                menuData.Text = uxName.Text != string.Empty ? uxName.Text :
menuData.Text;
                menuData.Description = uxDescription.Text != string.Empty ?
uxDescription.Text : menuData.Description;

                menuManager.Update(menuData);

                MessageUtilities.UpdateMessage(uxMessage, "Menu Id " + menuData.Id + "
has been updated", Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please enter valid Menu ID.",
Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update (menu item)

```
Update(Ektron.Cms.Organization.MenuItemData)
```

Updates an existing menu item in the CMS.

Authenticated users

- CMS Administrators
- Menu Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- *Content Menu Item ID
- Content Item ID
- Menu (Submenu) ID
- Description

Parameters

- `menuItemData`. The MenuItemData.

Remarks

Validate the menu item with `GetItem()` before you update the properties. Then, call `Update` with your modified MenuItemData object.

NOTE: You cannot change all properties after the initial Add event, such as the FolderData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
                    <ektronUI:Label ID="uxIdLabel" AssociatedControlID="uxId"
CssClass="span-4 last"
                        runat="server" Text="*Id (MenuId/SubmenuId):" />
                    <ektronUI:TextField ID="uxId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
                        Text="0" />
                </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxItemIdLabel" AssociatedControlID="uxItemId"
CssClass="span-4 last"
            runat="server" Text="Item Id (ContentId):" />
        <ektronUI:TextField ID="uxItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxMenuIdLabel" AssociatedControlID="uxMenuId"
CssClass="span-4 last"
            runat="server" Text=" Id (Menu/Submenu):" />
        <ektronUI:TextField ID="uxMenuId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>

    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last"
            runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3" ID="uxDescription" CssClass="span-4"
runat="server"
            ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Message ID="uxStatus" DisplayMode="Error" runat="server"
Visible="false" />
    </li>
        <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
```

```
Required" />
    </li>
</ol>

<ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MenuManager menuManager = new MenuManager();

            long menuId = long.Parse(uxMenuId.Text);
            long itemId = long.Parse(uxItemId.Text);
                       MenuItemData menuItemData = menuManager.GetMenuItem(long.Parse(uxId.Text));
            if (menuItemData == null)
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
MenuItemID.", Message.DisplayModes.Error);
                return;
            }
            if (!string.IsNullOrEmpty(uxMenuId.Text) && menuId > 0)
            {
                MenuData menuData = menuManager.GetMenu(menuId);
                if (menuData == null)
                {
                       uxStatus.Visible = true;
                       MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Menu ID.",
Message.DisplayModes.Error);
                       return;
                }

               menuItemData.ParentId = menuData.Id;
            }

            if (!string.IsNullOrEmpty(uxItemId.Text) && itemId > 0)
            {
```

```
                ContentManager contentManager = new ContentManager();
                ContentData cData = contentManager.GetItem(itemId);
                if (cData != null)
                {
                    menuItemData.ItemId = itemId;
                    menuItemData.Text = cData.Title;
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Content
ID.", Message.DisplayModes.Error);
                    return;
                }
            }

            menuItemData.Description = uxDescription.Text != string.Empty ?
uxDescription.Text : menuItemData.Description;
            menuManager.Update(menuItemData);

            MessageUtilities.UpdateMessage(uxMessage, "MenuItem Added with Id " +
menuItemData.Id, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update (submenu)

```
Update(Ektron.Cms.Organization.SubMenuData)
```

Updates menu data only for the submenu item in the CMS.

Authenticated users

- CMS Administrators
- Menu Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Submenu ID
- Title
- Description

Parameters

- `subMenuData`. The SubMenuData.

Remarks

Validate the menu item with `GetItem()` before you update the properties. Then, call `Update` with your modified SubMenuData object.

NOTE: You cannot change all properties after the initial Add event, such as the FolderData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxIdLabel" AssociatedControlID="uxId" CssClass="span-4 last"
            runat="server" Text="*Id(SubMenuID):" />
        <ektronUI:TextField ID="uxId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-4
last"
            runat="server" Text="Title:" />
        <ektronUI:TextField ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last"
            runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3" ID="uxDescription" CssClass="span-4"
runat="server"
            ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Message ID="uxStatus" DisplayMode="Error" runat="server"
Visible="false" />
    </li>
        <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Organization;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            MenuManager menuManager = new MenuManager();

            long submenuId = long.Parse(uxId.Text);

            SubMenuData submenuData = menuManager.GetMenu(long.Parse(uxId.Text)) as
SubMenuData;
            if (submenuData == null)
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
SubMenuID.", Message.DisplayModes.Error);
                return;
            }

            submenuData.Text = uxName.Text != string.Empty ? uxName.Text :
submenuData.Text;
            submenuData.Description = uxDescription.Text != string.Empty ?
uxDescription.Text : submenuData.Description;

            menuManager.Update(submenuData);
            MessageUtilities.UpdateMessage(uxMessage, "SubMenu with Id " +
submenuData.Id + " has been updated.", Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

MenuCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Organization
```

Properties

- `MenuCriteria()`

  ```
  public MenuCriteria()
  ```

- `MenuCriteria(Ektron.Cms.Organization.MenuProperty, EkEnumeration.OrderByDirection)`

  ```
  public MenuCriteria(Ektron.Cms.Organization.MenuProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `MenuProperty` are:

  - AssociatedFolders
  - AssociatedTemplates
  - Description
  - Href
  - Id
  - ImageOverride
  - LanguageId
  - Text
  - Type

MenuData

Namespace

```
Ektron.Cms.Common
```

Properties

- `AncestorId`

  ```
  Public AncestorID As Long
  ```

- `AssociatedFolders`

  ```
  Public AssociatedFolders As String
  ```

- `AssociatedTemplates`

  ```
  Public AssociatedTemplates As String
  ```

- `Clone() As Object`

  ```
  Public Function Clone() As Object Implements ICloneable.Clone
  ```

- `Description`

  ```
  Public Description As String
  ```

- FolderId

```
Public FolderID As Long
```

- HasChildren

```
Public HasChildren As Boolean
```

- ID

```
Public ID As Long
```

- Image

```
Public Image As String
```

- ImageOverride

```
Public ImageOverride As Boolean
```

- Item

```
Public Item() As Ektron.Cms.Common.MenuItemData
```

- Link

```
Public Link As String
```

- ParentID

```
Public ParentID As Long
```

- Template

```
Public Template As String
```

- Title

```
Public Title As String
```

- Type

```
Public Type As Ektron.Cms.Common.EkEnumeration.CMSMenuItemType
```

MenuItemCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Organization
```

Properties

- MenuItemCriteria()

```
public MenuItemCriteria()
```

- MenuItemCriteria(Ektron.Cms.Organization.MenuItemProperty, EkEnumeration.OrderByDirection)

```
public MenuItemCriteria(Ektron.Cms.Organization.MenuItemProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

  Accepted fields for the `MenuItemProperty` are:

  - ItemDescription
  - Itemid
  - ItemType

- LanguageId
- MenuId
- MenuItemId
- MenuName
- Title

MenuItemData

Namespace

```
Ektron.Cms.Common
```

Properties

- `Clone() As Object`

```
Public Function Clone() As Object Implements ICloneable.Clone
```

- `ItemContentSubType`

```
Public ItemContentSubType As Integer
```

- `ItemDescription`

```
Public ItemDescription As String
```

- `ItemID`

```
Public ItemID As Long
```

- `ItemImage`

```
Public ItemImage As String
```

- `ItemImageOverride`

```
Public ItemImageOverride As Boolean
```

- `ItemLink`

```
Public ItemLink As String
```

- `ItemQuickLink`

```
Public ItemQuickLink As String
```

- `ItemSubType`

```
Public ItemSubType As Integer
```

- `ItemTarget`

```
Public ItemTarget As String
```

- `ItemTitle`

```
Public ItemTitle As String
```

- `ItemType`

```
Public ItemType As Ektron.Cms.Common.EkEnumeration.CMSMenuItemType
```

- `Menu`

```
Public Menu As Ektron.Cms.Common.MenuData
```

SubMenuData

Namespace

```
Ektron.Cms.Organization
```

Properties

- `ItemId`. Gets or sets the item id.

  ```
  public long ItemId { set; get; }
  ```

- `ParentId`. Gets or sets the ID of the parent menu. Root menus will have a parent ID of 0.

  ```
  public override long ParentId { set; get; }
  ```

- `SubMenuData()`. Initializes a new instance of the SubMenuData class.

  ```
  public SubMenuData()
  ```

- `SubMenuData(string)`. Initializes a new instance of the SubMenuData class.

  ```
  public SubMenuData(string text)
  ```

TaxonomyItemManager

8.50 and higher

The TaxonomyItemManager class manages taxonomy items. For information about taxonomies, see Organizing Content with Taxonomies in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Organization.TaxonomyItemManager
```

Constructors

- `TaxonomyItemManager()`
- `TaxonomyItemManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `TaxonomyService`. TaxonomyService.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 1297
- *GetList* on page 1300
- `Reorder(Int64,Int64,Int32,TaxonomyReorderItemType,Boolean)`. 8.60 and higher Reorders the taxonomy items.
- *Update* on page 1303

Add

```
Add(Ektron.Cms.TaxonomyItemData)
```

Adds a taxonomy item from TaxonomyItemData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Taxonomy ID
- * Item Type
- * Item ID

Parameters

- `taxonomyData`. The TaxonomyItemData object to add.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIdLabel" AssociatedControlID="uxTaxonomyId"
CssClass="span-3 last" runat="server" Text="* Taxonomy Id:" />
        <ektronUI:TextField ID="uxTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxItemTypeLabel" AssociatedControlID="uxItemType"
CssClass="span-3 last" runat="server" Text="* Item Type:" />
        <asp:DropDownList ID="uxItemType" runat="server">
            <asp:ListItem Value="0">Content</asp:ListItem>
            <asp:ListItem Value="1">User</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyItemIdLabel"
AssociatedControlID="uxTaxonomyItemId" CssClass="span-3 last" runat="server" Text="*
Item Id:" />
        <ektronUI:TextField ID="uxTaxonomyItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
        <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
```

```
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TaxonomyItemManager taxonomyItemManager = new TaxonomyItemManager();
            long itemID = long.Parse(uxTaxonomyItemId.Text);
            long taxonomyID = long.Parse(uxTaxonomyId.Text);

            TaxonomyManager taxonomyManager = new TaxonomyManager();
            TaxonomyData taxonomyData = taxonomyManager.GetItem(taxonomyID);
            bool add = false;

            if (taxonomyData != null)
            {
                if (uxItemType.SelectedItem.ToString() == "Content")
                {
                    ContentManager contentManager = new ContentManager();
                    ContentData cData = contentManager.GetItem(itemID, false);

                    if (cData != null)
                    {
                        add = true;
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Item ID.", Message.DisplayModes.Error);
                        return;
                    }
                }
                else
                {
                    UserManager Usermanager = new UserManager();
                    //Check the UserId valid or not
```

```
                    UserData Userdata = Usermanager.GetItem(itemID);
                    if (Userdata != null)
                    {
                        add = true;
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Item ID.", Message.DisplayModes.Error);
                        return;
                    }
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Taxonomy
ID.", Message.DisplayModes.Error);
                return;
            }
            if (add)
            {
                TaxonomyItemData taxonomyItemData = new TaxonomyItemData()
                            {
                                TaxonomyId = taxonomyID,
                                ItemType =
(Ektron.Cms.Common.EkEnumeration.TaxonomyItemType)(Convert.ToInt32
(uxItemType.SelectedValue)),
                                ItemId = itemID
                            };

                taxonomyItemManager.Add(taxonomyItemData);
                MessageUtilities.UpdateMessage(uxMessage,  "Taxonomy Item Added. ",
Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a taxonomy item from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Taxonomy ID
- * Taxonomy Item ID
- * Item Type

Parameters

- `taxonomyId`. The taxonomy ID of the item to be deleted.
- `taxonomyItemId`. The taxonomy item ID to be deleted.
- `type`. The item taxonomy type to be deleted.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIDLabel" AssociatedControlID="uxTaxonomyID"
CssClass="span-4 last" runat="server" Text="* Taxonomy Id:" />
        <ektronUI:TextField ID="uxTaxonomyID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyItemIDLabel"
AssociatedControlID="uxTaxonomyItemID" CssClass="span-4 last" runat="server" Text="*
Taxonomy Item Id:" />
        <ektronUI:TextField ID="uxTaxonomyItemID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxItemTypeLabel" AssociatedControlID="uxItemType"
CssClass="span-4 last" runat="server" Text="* Item Type:" />
        <asp:DropDownList ID="uxItemType" runat="server">
            <asp:ListItem>Content</asp:ListItem>
            <asp:ListItem>User</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
```

```
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TaxonomyItemManager taxonomyItemManager = new TaxonomyItemManager();
            TaxonomyManager taxonomyManager = new TaxonomyManager();

            long taxonomyID = long.Parse(uxTaxonomyID.Text);
            long itemID = long.Parse(uxTaxonomyItemID.Text);
            EkEnumeration.TaxonomyItemType taxonomyItemType;

            TaxonomyData taxonomyData = taxonomyManager.GetItem(taxonomyID);
            bool delete = false;

            if (taxonomyData != null)
            {
                if (uxItemType.SelectedItem.ToString() == "Content")
                {
                    taxonomyItemType = EkEnumeration.TaxonomyItemType.Content;
                    ContentManager contentManager = new ContentManager();
                    ContentData cData = contentManager.GetItem(itemID, false);

                    if (cData != null)
                    {
                        delete = true;
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Item ID.", Message.DisplayModes.Error);
                        return;
                    }
                }
                else
```

```
                {
                    taxonomyItemType = EkEnumeration.TaxonomyItemType.User;
                    UserManager Usermanager = new UserManager();
                    //Check the UserId valid or not
                    UserData Userdata = Usermanager.GetItem(itemID);
                    if (Userdata != null)
                    {
                        delete = true;
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Item ID.", Message.DisplayModes.Error);
                        return;
                    }
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Taxonomy
ID.", Message.DisplayModes.Error);
                return;
            }
            if (delete)
            {
                taxonomyItemManager.Delete(taxonomyID, itemID, taxonomyItemType);
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy Item with Id " +
itemID.ToString() + " deleted.", Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(TaxonomyItemCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Taxonomy Item Property
- * Object Value

Parameters

- `criteria`. The TaxonomyItemCriteria to filter taxonomy items on.

Returns

List with the retrieved TaxonomyItemData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyItemPropertyLabel"
AssociatedControlID="uxTaxonomyItemProperty" CssClass="span-5 last" runat="server"
Text="Taxonomy Item Property:" />
        <asp:DropDownList ID="uxTaxonomyItemProperty" runat="server">
            <asp:ListItem>Taxonomy Id</asp:ListItem>
            <asp:ListItem>Taxonomy Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last" runat="server" Text="* Object Value:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="189" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxTaxonomyItemlistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Taxonomy Item Title
                    </th>
                    <th>
                        Item Id
                    </th>
                    <th>
                        Taxonomy Item Type
                    </th>
```

```
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("TaxonomyItemTitle")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ItemId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TaxonomyItemType")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Organization;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            TaxonomyItemCriteria criteria = new TaxonomyItemCriteria();
            if (uxTaxonomyItemProperty.SelectedItem.Text == "Taxonomy Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(TaxonomyItemProperty.TaxonomyId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxonomyItemProperty.TaxonomyName,
CriteriaFilterOperator.EqualTo, Objectvalue);
```

```
            }

            TaxonomyItemManager taxonomyItemManager = new TaxonomyItemManager();
            List<TaxonomyItemData> taxonomyItemList = taxonomyItemManager.GetList
(criteria);

            uxTaxonomyItemlistView.Visible = true;
            uxTaxonomyItemlistView.DataSource = taxonomyItemList;
            uxTaxonomyItemlistView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.FolderData)
```

Updates an existing TaxonomyItemData in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Taxonomy ID
- * Item Type
- * Item ID

Parameters

- `taxonomyData`. The TaxonomyItemData object to update.

Returns

Returns the custom CmsData object updated.

Remarks

Validate the Taxonomy Item item with `GetItem()` before you update the properties. Then, call `Update` with your modified TaxonomyItemData object.

NOTE: You cannot change all properties after the initial Add event, such as the TaxonomyItemData.Type.

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxTaxonomyIdLabel" AssociatedControlID="uxTaxonomyId"
CssClass="span-3 last" runat="server" Text="* Taxonomy Id:" />
            <ektronUI:TextField ID="uxTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
        </li>
        <li class="clearfix">
            <ektronUI:Label ID="uxItemTypeLabel" AssociatedControlID="uxItemType"
CssClass="span-3 last" runat="server" Text="* Item Type:" />
            <asp:DropDownList ID="uxItemType" runat="server">
                <asp:ListItem Value="0">Content</asp:ListItem>
                <asp:ListItem Value="1">User</asp:ListItem>
            </asp:DropDownList>
        </li>
        <li class="clearfix">
            <ektronUI:Label ID="uxTaxonomyItemIdLabel"
AssociatedControlID="uxTaxonomyItemId" CssClass="span-3 last" runat="server" Text="*
Item Id:" />
            <ektronUI:TextField ID="uxTaxonomyItemId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
        </li>
            <li class="clearfix" style="color: red;">
                    <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                        Text="Status:" />
                </li>
        <li class="clearfix">
            <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
            <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server"
Text="* - Required" />
        </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            TaxonomyItemManager taxonomyItemManager = new TaxonomyItemManager();
            long itemID = long.Parse(uxTaxonomyItemId.Text);
            long taxonomyID = long.Parse(uxTaxonomyId.Text);

            TaxonomyManager taxonomyManager = new TaxonomyManager();
            TaxonomyData taxonomyData = taxonomyManager.GetItem(taxonomyID);
            bool update = false;

            if (taxonomyData != null)
            {
                if (uxItemType.SelectedItem.ToString() == "Content")
                {
                    ContentManager contentManager = new ContentManager();
                    ContentData cData = contentManager.GetItem(itemID, false);

                    if (cData != null)
                    {
                        update = true;
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus,  "Please Enter Valid
Item ID.", Message.DisplayModes.Error);
                        return;
                    }
                }
                else
                {
                    UserManager Usermanager = new UserManager();
                    //Check the UserId valid or not
                    UserData Userdata = Usermanager.GetItem(itemID);
                    if (Userdata != null)
                    {
                        update = true;
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Item ID.", Message.DisplayModes.Error);
                        return;
                    }
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Taxonomy
ID.", Message.DisplayModes.Error);
                return;
            }
            if (update)
```

```
            {
                TaxonomyItemData taxonomyItemData = new TaxonomyItemData()
                {
                    TaxonomyId = taxonomyID,
                    ItemType = (Ektron.Cms.Common.EkEnumeration.TaxonomyItemType)
(Convert.ToInt32(uxItemType.SelectedValue)),
                    ItemId = itemID
                };

                taxonomyItemManager.Update(taxonomyItemData);
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy Item Updated. ",
Message.DisplayModes.Success);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

TaxonomyItemCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Organization
```

Properties

- `ReturnRecursiveChildren`. Gets or sets the `ReturnRecursiveChildren` flag. If true AND a `ParentId` filter is supplied, all recursive children taxonomies will be returned for the parent ID.

  ```
  public bool ReturnRecursiveChildren { set; get; }
  ```

- `TaxonomyItemCriteria()`

  ```
  public TaxonomyItemCriteria()
  ```

- `TaxonomyItemCriteria (Ektron.Cms.Organization.TaxonomyItemProperty, EkEnumeration.OrderByDirection)`

  ```
  public TaxonomyItemCriteria(Ektron.Cms.Organization.TaxonomyItemProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `TaxonomyItemProperty` are:

  - AssignedByUserId
  - DateCreated

- DateModified
- ItemId
- ItemType
- LanguageId
- TaxonomyId
- TaxonomyItemDisplayOrder
- TaxonomyLanguageId
- TaxonomyName
- TaxonomyPath
- TaxonomyType
- Title

TaxonomyItemData

Namespace

```
Ektron.Cms
```

Properties

- AddByUserId

```
public long AddedByUserId { set; get; }
```

- AnalyticsInfo

```
public Ektron.Cms.Controls.CmsWebService.AnalyticsData
  AnalyticsInfo { set; get; }
```

- AssetData

```
public Ektron.Cms.Controls.CmsWebService.AssetData
  AssetData { set; get; }
```

- AssetInfo

```
public Ektron.Cms.Controls.CmsWebService.AssetData
  AssetInfo { set; get; }
```

- ContentSubType

```
public Ektron.Cms.Controls.CmsWebService.CMSContentSubtype
  ContentSubType { set; get; }
```

- ContentType

```
public string ContentType { set; get; }
```

- DateCreated

```
public System.DateTime DateCreated { set; get; }
```

- DateModified

```
public System.DateTime DateModified { set; get; }
```

- EditorLastName

```
public string EditorLastName { set; get; }
```

- EndDate

```
public System.DateTime EndDate { set; get; }
```

- FilePath

```
public string FilePath { set; get; }
```

- FirstName

```
public string FirstName { set; get; }
```

- FolderId

```
public long FolderId { set; get; }
```

- GoLiveDate

```
public System.DateTime GoLiveDate { set; get; }
```

- Html

```
public string Html { set; get; }
```

- Image

```
public string Image { set; get; }
```

- ItemId

```
public long ItemId { set; get; }
```

- ItemLanguageId

```
public int ItemLanguageId { set; get; }
```

- ItemType

```
public Ektron.Cms.Controls.CmsWebService.TaxonomyItemType
  ItemType { set; get; }
```

- ObjectType

```
public int ObjectType { set; get; }
```

- QuickLink

```
public string QuickLink { set; get; }
```

- Status

```
public Ektron.Cms.Controls.CmsWebService.ItemStatus
  Status { set; get; }
```

- TaxonomyItemDisplayDateCreated

```
public string TaxonomyItemDisplayDateCreated { set; get; }
```

- TaxonomyItemDisplayDateModified

```
public string TaxonomyItemDisplayDateModified { set; get; }
```

- TaxonomyItemStatus

```
public Ektron.Cms.Controls.CmsWebService.ItemStatus
  TaxonomyItemStatus { set; get; }
```

- TaxonomyItemTitle

```
public string TaxonomyItemTitle { set; get; }
```

- TaxonomyLanguageId

```
public int TaxonomyLanguageId { set; get; }
```

- Teaser

```
public string Teaser { set; get; }
```

- Thumbnail

```
public string Thumbnail { set; get; }
```

TaxonomyManager

8.50 and higher

The TaxonomyManager class manages taxonomies. For information about taxonomies, see Organizing Content with Taxonomies in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Organization.TaxonomyManager
```

Constructors

- `TaxonomyManager()`
- `TaxonomyManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `TaxonomyService`. Taxonomy Service
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* on the facing page
- *Delete* on page 1313
- *GetItem (taxonomy ID)* on page 1315
- *GetItem (taxonomy path)* on page 1317
- *GetList* on page 1319
- `GetList(TaxonomyCustomPropertyCriteria)`. 8.60 and higher Returns a list of all taxonomies based upon custom property criteria supplied.
- *GetTaxonomyConfigurationList* on page 1321

- `GetTree(Int64,Int32,Boolean,PagingInfo,TaxonomyType,TaxonomyItemsSortOrder)`. Returns a taxonomy tree based on taxonomy ID.
- `GetTree(String,Int32,Boolean,PagingInfo,TaxonomyType,TaxonomyItemsSortOrder)`. Returns a taxonomy tree based on taxonomy path.
- `ImportTaxonomy(String,String)`. 8.60 and higher Creates a taxonomy using XML file.
- `MoveTaxonomy(Int64,Int64)`. 8.60 and higher Moves the taxonomy with items from one to another at any level.
- `Reorder(Int64,Int32,Boolean)`. 8.60 and higher Reorders the taxonomy category based on taxonomy ID.
- *Update* on page 1323
- *UpdateTaxonomyConfigurations* on page 1326

Add

```
Add(Ektron.Cms.TaxonomyData)
```

Adds a taxonomy from TaxonomyData object.

Authenticated users

- CMS Administrators
- Taxonomy Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Name
- * Parent ID
- Taxonomy Type
- Description

Parameters

- `taxonomyData`. The TaxonomyData to add.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyNameLabel" AssociatedControlID="uxTaxonomyName"
CssClass="span-3 last" runat="server" Text="* Name:" />
        <ektronUI:TextField ID="uxTaxonomyName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxParentIdLabel" AssociatedControlID="uxParentId"
CssClass="span-3 last" runat="server" Text="* Parent Id:" />
        <ektronUI:TextField ID="uxParentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyTypeLabel" AssociatedControlID="uxTaxonomyType"
CssClass="span-3 last" runat="server" Text=" Taxonomy Type:" />
        <asp:DropDownList ID="uxTaxonomyType" runat="server">
            <asp:ListItem Value="0">Content</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-3 last" runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" ID="uxDescription" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup"  />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {

        try
```

```
        {
            TaxonomyManager taxonomyManager = new TaxonomyManager();
            TaxonomyData taxonomyData = new TaxonomyData()
            {
                Name = uxTaxonomyName.Text,
                ParentId = long.Parse(uxParentId.Text),
                Description = uxDescription.Text,
                Visible = true,
                TaxonomyType = (Ektron.Cms.Common.EkEnumeration.TaxonomyType)
(Convert.ToInt32(uxTaxonomyType.Text)),
            };

            taxonomyData.TaxonomyConfigurations.Add
(Ektron.Cms.Common.EkEnumeration.TaxonomyConfiguration.Content);
            taxonomyManager.Add(taxonomyData);

            MessageUtilities.UpdateMessage(uxMessage, "Taxonomy Added with Id  " +
taxonomyData.Id, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a taxonomy from the CMS.

Authenticated users

- CMS Administrators
- Taxonomy Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `taxonomyId`

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIdLabel" AssociatedControlID="uxTaxonomyId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxonomyID = long.Parse(uxTaxonomyId.Text);
            TaxonomyManager taxonomyManager = new TaxonomyManager();
            TaxonomyData taxonomyData = taxonomyManager.GetItem(taxonomyID);
            if (taxonomyData != null)
            {
                taxonomyManager.Delete(taxonomyID);
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy with Id " +
taxonomyID.ToString() + " deleted", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy with Id " +
taxonomyID.ToString() + " is Invalid", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

```
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem (taxonomy ID)

```
GetItem(System.Int64)
```

Retrieves a taxonomy node-based on taxonomy ID. To retrieve child items, use `GetTree()`.

Authenticated users

- CMS Administrators
- Taxonomy Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of taxonomy node to return.

Returns

TaxonomyData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIdLabel" AssociatedControlID="uxTaxonomyId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
```

```
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLevel" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxVisible" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxonomyId = long.Parse(uxTaxonomyId.Text);

            TaxonomyManager taxonomyManager = new TaxonomyManager();
            TaxonomyData taxonomyData = taxonomyManager.GetItem(taxonomyId);
           // taxonomyData = taxonomyManager.GetItem(taxonomyId);

            if (taxonomyData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for taxonomy with ID
" + taxonomyData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxName.Text = "Name : " + taxonomyData.Name;
                uxType.Text = "Taxonomy Type : " + taxonomyData.TaxonomyType;
                uxDescription.Text = "Description :" + taxonomyData.Description;
                uxLevel.Text = "Level :" + taxonomyData.Level;
                uxVisible.Text = "Visible :" + taxonomyData.Visible;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,"Taxonomy does not exists. ",
```

```
Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem (taxonomy path)

```
GetItem(System.String)
```

Retrieves a taxonomy node-based on taxonomy path. To retrieve child items, use `GetTree()`.

Authenticated users

- CMS Administrators
- Taxonomy Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Taxonomy Path

Parameters

- `path`. Path of taxonomy node to return.

Returns

TaxonomyData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyPathLabel" AssociatedControlID="uxTaxonomyPath"
CssClass="span-3 last" runat="server" Text="* Taxonomy Path :" />
        <ektronUI:TextField ID="uxTaxonomyPath" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="\" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
```

```
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLevel" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxVisible" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TaxonomyManager taxonomyManager = new TaxonomyManager();
            TaxonomyData taxonomyData = taxonomyManager.GetItem(uxTaxonomyPath.Text);

            if (taxonomyData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for taxonomy with ID
" + taxonomyData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxName.Text = "Name : " + taxonomyData.Name;
                uxType.Text = "Taxonomy Type : " + taxonomyData.TaxonomyType;
                uxDescription.Text = "Description :" + taxonomyData.Description;
                uxLevel.Text = "Level :" + taxonomyData.Level;
                uxVisible.Text = "Visible :" + taxonomyData.Visible;
            }
            else
```

```
            {
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy does not exists. ",
Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Organization.TaxonomyCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators
- Taxonomy Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Taxonomy Property
- * Object Value

Parameters

- `criteria`. TaxonomyCriteria on which to base the filter.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyPropertyLabel"
AssociatedControlID="uxTaxonomyProperty" CssClass="span-5 last" runat="server"
Text="Taxonomy Property:" />
        <asp:DropDownList ID="uxTaxonomyProperty" runat="server">
            <asp:ListItem>Taxonomy Id</asp:ListItem>
            <asp:ListItem>Taxonomy Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
```

```
CssClass="span-5 last" runat="server" Text="* Object Value:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="189" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxTaxonomylistView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Taxonomy Name
                    </th>
                    <th>
                        Taxonomy Path
                    </th>
                    <th>
                        Taxonomy Level
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("TaxonomyName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TaxonomyPath")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TaxonomyLevel")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Organization;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            TaxonomyCriteria criteria = new TaxonomyCriteria();
            if (uxTaxonomyProperty.SelectedItem.Text == "Taxonomy Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter( TaxonomyProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaxonomyProperty.Name,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            TaxonomyManager taxonomyManager = new TaxonomyManager();
            List<TaxonomyData> taxonomyItemList = taxonomyManager.GetList(criteria);

            uxTaxonomylistView.Visible = true;
            uxTaxonomylistView.DataSource = taxonomyItemList;
            uxTaxonomylistView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTaxonomyConfigurationList

```
GetTaxonomyConfigurationList(System.Int64)
```

Retrieves a list of taxonomy configurations for a taxonomy. Taxonomy configurations determine the type of CMS objects that you can assign to a taxonomy.

Authenticated users

- CMS Administrators
- Taxonomy Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `taxonomyId`. ID of taxonomy to retrieve configurations for.

Returns

List of taxonomy configuration enumerations.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIdLabel" AssociatedControlID="uxTaxonomyId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetConfigurationList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxonomyId = long.Parse(uxTaxonomyId.Text);

            TaxonomyManager taxonomyManager = new TaxonomyManager();
            TaxonomyData taxonomyData = taxonomyManager.GetItem(taxonomyId);

            if (taxonomyData != null)
            {
                List<EkEnumeration.TaxonomyConfiguration> list =
taxonomyManager.GetTaxonomyConfigurationList(taxonomyId);

                if (list != null)
                {
                    MessageUtilities.UpdateMessage(uxMessage,  "Taxonomy Configuration
for taxonomy ID " + taxonomyId + " is " + list[0].ToString() + " .",
Message.DisplayModes.Success);
                }

            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
TaxonomyId.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
}
```

Update

```
Update(Ektron.Cms.TaxonomyData)
```

Update is used to update an existing TaxonomyData object in the CMS.

Authenticated users

- CMS Administrators
- Taxonomy Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- Name
- Parent ID
- * Taxonomy Type
- Description

Parameters

- `taxonomyData`

Returns

Returns the custom CmsData object updated.

Remarks

Validate the Taxonomy item with `GetItem()` before you update the properties. Then, call `Update` with your modified TaxonomyData object.

NOTE: You cannot change all properties after the initial Add event, such as the TaxonomyData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIdLabel" AssociatedControlID="uxTaxonomyId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyNameLabel" AssociatedControlID="uxTaxonomyName"
CssClass="span-3 last" runat="server" Text=" Name:" />
        <ektronUI:TextField ID="uxTaxonomyName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxParentIdLabel" AssociatedControlID="uxParentId"
CssClass="span-3 last" runat="server" Text=" Parent Id:" />
        <ektronUI:TextField ID="uxParentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyTypeLabel" AssociatedControlID="uxTaxonomyType"
CssClass="span-3 last" runat="server" Text=" * Taxonomy Type:" />
        <asp:DropDownList ID="uxTaxonomyType" runat="server">
            <asp:ListItem Value="0">Content</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-3 last" runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" ID="uxDescription" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxonomyId = long.Parse(uxTaxonomyId.Text);
            TaxonomyManager taxonomyManager = new TaxonomyManager();
            TaxonomyData taxonomyData = taxonomyManager.GetItem(taxonomyId);

            if (taxonomyData != null)
            {
                taxonomyData.Name = (uxTaxonomyName.Text != string.Empty ?
uxTaxonomyName.Text : taxonomyData.Name);
                taxonomyData.TaxonomyType = (EkEnumeration.TaxonomyType )Convert.ToInt32
(uxTaxonomyType.SelectedItem.Value);
                taxonomyData.ParentId = (uxParentId.Text != string.Empty ? long.Parse
(uxParentId.Text) : taxonomyData.ParentId);
                taxonomyData.Description = (uxDescription.Text != string.Empty
?uxDescription.Text :taxonomyData.Description);
```

```
                taxonomyManager.Update(taxonomyData);
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy updated. ",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy does not exists. ",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

UpdateTaxonomyConfigurations

```
UpdateTaxonomyConfigurations(System.Int64,System.Collections.Generic.List)
```

Updates a configuration list for a taxonomy.

Authenticated users

- CMS Administrators
- Taxonomy Administrators
- Users that have access permission

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- Taxonomy Configuration

Parameters

- `taxonomyId`. Taxonomy ID that is updated.
- `configurationList`. List of taxonomy configurations to apply for update.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIdLabel" AssociatedControlID="uxTaxonomyId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxTaxonomyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
```

```
        <li class="clearfix">
                <ektronUI:Label ID="uxTaxonomyConfigurationLabel"
AssociatedControlID="uxTaxonomyConfiguration" CssClass="span-5 last" runat="server"
Text="Taxonomy Configuration:" />
                <asp:DropDownList ID="uxTaxonomyConfiguration" runat="server">
                    <asp:ListItem>Content</asp:ListItem>
                    <asp:ListItem>User</asp:ListItem>
                    <asp:ListItem>Group</asp:ListItem>
                </asp:DropDownList>
            </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="UpdateConfigurationList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxonomyId = long.Parse(uxTaxonomyId.Text);

            TaxonomyManager taxonomyManager = new TaxonomyManager();
            TaxonomyData taxonomyData = taxonomyManager.GetItem(taxonomyId);

            if (taxonomyData != null)
            {
                List<EkEnumeration.TaxonomyConfiguration> list = new
List<EkEnumeration.TaxonomyConfiguration>();
```

```
                if (uxTaxonomyConfiguration.SelectedItem.ToString() == "Content")
                {
                    list.Add(EkEnumeration.TaxonomyConfiguration.Content);
                }
                else if (uxTaxonomyConfiguration.SelectedItem.ToString() == "User")
                {
                    list.Add(EkEnumeration.TaxonomyConfiguration.User);
                }
                else
                {
                    list.Add(EkEnumeration.TaxonomyConfiguration.Group);
                }
                taxonomyManager.UpdateTaxonomyConfigurations(taxonomyId, list);
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy Configuration for
taxonomy ID " + taxonomyId + " has updated.", Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
TaxonomyId.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }c
```

Data Classes

TaxonomyCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Organization
```

Properties

- `ReturnRecursiveChildren`. Gets or sets the `ReturnRecursiveChildren` flag. If true AND a `ParentId` filter is supplied, all recursive children taxonomies will be returned for the parent ID.

  ```
  public bool ReturnRecursiveChildren { set; get; }
  ```

- `TaxonomyCriteria()`

  ```
  public TaxonomyCriteria()
  ```

- `TaxonomyCriteria(Ektron.Cms.Organization.TaxonomyProperty, EkEnumeration.OrderByDirection)`

```
public TaxonomyCriteria(Ektron.Cms.Organization.TaxonomyProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `TaxonomyProperty` are:

- DateCreated
- HasChildren
- Id
- ItemCount
- LanguageId
- Level
- Name
- ParendId
- Path
- TaxonomyDisplayOrder
- TemplateId
- Type

TaxonomyData

Namespace

```
Ektron.Cms
```

Properties

- `FolderId`

  ```
  public long FolderId { set; get; }
  ```

- `Taxonomy`

  ```
  public Ektron.Cms.Controls.CmsWebService.TaxonomyData[]
    Taxonomy { set; get; }
  ```

- `TaxonomyConfigurations`

  ```
  public Ektron.Cms.Controls.CmsWebService.TaxonomyConfiguration[]
    TaxonomyConfigurations { set; get; }
  ```

- `TaxonomyItems`

  ```
  public Ektron.Cms.Controls.CmsWebService.TaxonomyItemData[]
    TaxonomyItems { set; get; }
  ```

Search

The Search manager category manages online searches and has the following classes:

- *SearchManager* on the facing page. Provides access to query the index of a configured search provider.
- *QueryPropositionManager* on page 1341. 9.00 and higher Provides access to query the index of a configured search provider.
- *Search Data Classes* on page 1346
- *Search Exceptions* on page 1384
- *Search Expressions* on page 1389
- *Search Abstract Classes and Interfaces* on page 1412

SearchManager

The SearchManager provides access to query the index of a configured search provider.

Namespace

```
Ektron.Cms.Framework.Search.SearchManager
```

Constructors

- `SearchManager()`
- `SearchManager(ApiAccessMode)`

Properties

- `SearchService`. Gets the underlying service facilitating access to the configured search provider.

Methods

- *Search(AdvancedSearchCriteria)* on the next page
- *Search(KeywordSearchCriteria)* on page 1336

Remarks

- `All Providers`. For diagnostic purposes, the search provider's interpretation of your search criteria can be logged to the Windows Application log. This represents the raw query issued to the search engine. To log this information, set the value of the `LogLevel` application setting in your site's `web.config` to `3`.
- `FAST Search for Sharepoint 2010`. The AdvancedSearchCriteria is not applicable when your search provider is Microsoft FAST Search for SharePoint. For the purposes of easing migration to this provider, any queries using this criteria is interpreted as KeywordSearchCriteria.
- `Search Server 2010`. MSDN states that the maximum length for a keyword query is 2048 characters. If you encounter this limitation using the KeywordSearchCriteria, consider revising your query or switching to the AdvancedSearchCriteria which supports longer queries.

  MSDN also states that the maximum length for a full-text search query is 4096 characters. If you encounter this limitation using the AdvancedSearchCriteria, consider revising your query.
- `Solr`. The AdvancedSearchCriteria is not applicable when your search provider is Apache Solr. For the purposes of easing migration to this provider, any queries using this criteria is interpretedas KeywordSearchCriteria.

  By default, the content of assets (Word documents, PDFs, and so on) is indexed for files up to 10 MB in size. This limit is intended to accelerate crawl times and

minimize the burden placed upon the search server by the content extraction process. When a file exceeds this limit, its summary is indexed instead of the full content of the document. The limit can be adjusted via the ManifoldCF administration console. (Use caution when updating advanced configuration options such as this. Changes may impact the performance of your search server.)

Solr relies on Apache Tika, an open source collection of content analysis libraries, for the extraction of content from asset files (Word documents, PDFs, and so on). Support for the extraction of content from these types of files is subject to the features and limitations of this toolkit.

Search(AdvancedSearchCriteria)

```
Search(Ektron.Cms.Search.KeywordSearchCriteria)
```

Issues advanced search to return specific results and run complex queries.

Submits a query according to the specified criteria. The AdvancedSearchCriteria is intended to target specific, narrow, sets of data. The length and complexity of queries supported by this method are subject to the limitations of the configured search provider. For some search providers, this method may also allow for longer or more complicated queries than may be allowed with the KeywordSearchCriteria. This method is not supported for all search providers.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Parameters

- `criteria`. Search criteria.

Returns

Search response data.

Return properties

You want the following properties to be returned when viewing the results.

```
criteria.ReturnProperties = new HashSet<PropertyExpression>()
{
  SearchContentProperty.Id,
  SearchContentProperty.Title,
  SearchContentProperty.QuickLink
};
```

OrderBy

Sets the order of search results based on a return property `Ascending/Descending`.

```
criteria.OrderBy = new List<OrderData>()
{
```

```
  new OrderData(SearchContentProperty.Rank, OrderDirection.Descending)
};
```

PagingInfo

Sets the number of results you see per page and the current page number for search.

```
criteria.PagingInfo = new PagingInfo(10);
criteria.PagingInfo.CurrentPage = 1;
```

ExpressionTree

Helps build nested or complex queries using managed properties. This examples shows how to search for the Content ID specified by the user in the currently active site language or default content language (if no active language is found). Also, search results are restricted by Content Type (`Ektron.Cms.Search.SearchType.IsNonUserContent()`) to ensure users or community groups do not appear.

```
ContentManager contentManager = new ContentManager();
Expression expressionTree = Ektron.Cms.Search.SearchType.IsNonUserContent();
if (contentManager.RequestInformation.ContentLanguage == 0)
  expressionTree &= SearchContentProperty.Language.EqualTo
(contentManager.RequestInformation.DefaultContentLanguage);
else if (contentManager.RequestInformation.ContentLanguage > 0)
  expressionTree &= SearchContentProperty.Language.EqualTo
(contentManager.RequestInformation.ContentLanguage);
  expressionTree &= SearchContentProperty.Id.EqualTo(id);
  criteria.ExpressionTree = expressionTree;
```

.aspx code snippet

```
<asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="0">
    <asp:View ID="View1" runat="server">
        <div class="ektron-ui-form registerForm">
            <ol class="formFields">
                <li class="clearfix">
                    <ektronUI:Label ID="Label1" AssociatedControlID="uxContentId"
CssClass="span-8 last"
                            runat="server" Text="Search for Content Id (in current site
language):" />
                        <ektronUI:TextField ID="TextField1" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup"
                            Text="30" />
                </li>
                </div>
                <li class="clearfix">
                    <ektronUI:Button ID="Button1" runat="server" OnClick="uxSubmit_
Click" Text="Search"></ektronUI:Button>
                </li>
            </ol>
        </div>
    </asp:View>
    <asp:View ID="View2" runat="server">
        <ektronUI:Message ID="Message1" DisplayMode="Information" Visible="false"
runat="server" />
```

```
        <asp:ListView ID="ListView1" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
            Visible="false">
            <EmptyDataTemplate>
                Search criteria did not match any records.
            </EmptyDataTemplate>
            <LayoutTemplate>
                <table class="devsite-api-method">
                    <thead>
                        <tr>
                            <th>
                                Content Id
                            </th>
                            <th>
                                Content Title
                            </th>
                            <th>
                                Quicklink
                            </th>
                        </tr>
                    </thead>
                    <tbody>
                        <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                    </tbody>
                </table>
            </LayoutTemplate>
            <ItemTemplate>
            <h1 > FOR C# ASPX <h1 />
                <tr>
                    <td class="devsite-method" style="width: 55px">
                        <%#((SearchResultData)Container.DataItem)
[SearchContentProperty.Id].ToString()%>
                    </td>
                    <td class="devsite-method">
                        <%# ((SearchResultData)Container.DataItem)
[SearchContentProperty.Title].ToString()%>
                    </td>
                    <td class="devsite-method">
                        <%# ((SearchResultData)Container.DataItem)
[SearchContentProperty.QuickLink].ToString()%>
                    </td>
                </tr>
                <h1 > FOR VB ASPX <h1 />
                    <tr>
                        <td class="devsite-method" style="width: 55px">
                            <%# Container.DataItem(SearchContentProperty.Id).ToString
()%>
                        </td>
                        <td class="devsite-method">
                            <%# Container.DataItem(SearchContentProperty.Title).ToString
()%>
                        </td>
                        <td class="devsite-method">
                            <%# Container.DataItem
(SearchContentProperty.QuickLink).ToString()%>
```

```
                        </td>
                    </tr>
                </ItemTemplate>
            </asp:ListView>
        </asp:View>
</asp:MultiView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms.Common;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.Search;
using Ektron.Cms.Search.Expressions;
using Ektron.Cms;
using Ektron.Cms.Search;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long id;
            if (long.TryParse(uxContentId.Text, out id))
            {
                AdvancedSearchCriteria criteria = new AdvancedSearchCriteria();
                criteria.OrderBy = new List<OrderData>()
                {
                    new OrderData(SearchContentProperty.Rank, OrderDirection.Descending)
                };
                criteria.PagingInfo = new PagingInfo(10);
                criteria.PagingInfo.CurrentPage = 1;
                criteria.ReturnProperties = new HashSet<PropertyExpression>()
                {
                    SearchContentProperty.Id,
                    SearchContentProperty.Title,
                    SearchContentProperty.QuickLink
                };

                ContentManager contentManager = new ContentManager();
                Expression expressionTree =
Ektron.Cms.Search.SearchType.IsNonUserContent();
                if (contentManager.RequestInformation.ContentLanguage == 0)
                    expressionTree &= SearchContentProperty.Language.EqualTo
(contentManager.RequestInformation.DefaultContentLanguage);
                else if (contentManager.RequestInformation.ContentLanguage > 0)
                    expressionTree &= SearchContentProperty.Language.EqualTo
```

```
(contentManager.RequestInformation.ContentLanguage);

                expressionTree &= SearchContentProperty.Id.EqualTo(id);
                criteria.ExpressionTree = expressionTree;

                ISearchManager manager = ObjectFactory.GetSearchManager();

                SearchResponseData response = manager.Search(criteria);
                uxSearchResultView.Visible = true;
                uxSearchResultView.DataSource = response.Results;
                uxSearchResultView.DataBind();

                uxPageMultiView.SetActiveView(uxViewMessage);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter a numeric
Content Id", Message.DisplayModes.Error);
                uxPageMultiView.SetActiveView(uxViewMessage);
            }
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Search(KeywordSearchCriteria)

```
Search(Ektron.Cms.Search.KeywordSearchCriteria)
```

Issues a simple text search query that you can combine with complex managed property queries.

Submits a query according to the specified criteria. The criteria may be a combination of free form search terms and programmatically generated query expressions. The length and complexity of queries supported by this method are subject to the limitations of the configured search provider.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Parameters

- `criteria`. Search criteria.

Returns

Search response data.

ReturnProperties

Returns the properties you want in the results.

```
criteria.ReturnProperties = new HashSet<PropertyExpression>()
{
  SearchContentProperty.Id,
  SearchContentProperty.Title,
  SearchContentProperty.QuickLink
};
```

OrderBy

Sets the order of search results based on a return property `Ascending/Descending`.

```
criteria.OrderBy = new List<OrderData>()
{
new OrderData(SearchContentProperty.Rank, OrderDirection.Descending)
};
```

PagingInfo

Sets the number of results you see per page and the current page number for search.

```
criteria.PagingInfo = new PagingInfo(10);
criteria.PagingInfo.CurrentPage = 1;
```

QueryText

Set the text search query.

```
criteria.QueryText = uxSearchText.Text;
```

ExpressionTree

Helps build nested or complex queries using managed properties. The following example shows how to search for the content in the currently active Site language or default content language (if no active language is found). Also, the search results are restricted by Content Type (`Ektron.Cms.Search.SearchType.IsNonUserContent()`) to ensure users or community groups do not appear.

```
ContentManager contentManager = new ContentManager();
Expression expressionTree = Ektron.Cms.Search.SearchType.IsNonUserContent();
if (contentManager.RequestInformation.ContentLanguage == 0)
  expressionTree &= SearchContentProperty.Language.EqualTo
(contentManager.RequestInformation.DefaultContentLanguage);
else if (contentManager.RequestInformation.ContentLanguage > 0)
  expressionTree &= SearchContentProperty.Language.EqualTo
(contentManager.RequestInformation.ContentLanguage);
```

.aspx code snippet

```
<asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="0">
    <asp:View ID="View1" runat="server">
        <div class="ektron-ui-form registerForm">
            <ol class="formFields">
                <li class="clearfix">
```

```
                    <ektronUI:Label ID="Label1" AssociatedControlID="uxContentId"
CssClass="span-8 last"
                            runat="server" Text="Search for Content Id (in current site
language):" />
                        <ektronUI:TextField ID="TextField1" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup"
                            Text="30" />
                </li>
                </div>
                <li class="clearfix">
                    <ektronUI:Button ID="Button1" runat="server" OnClick="uxSubmit_
Click" Text="Search"></ektronUI:Button>
                </li>
            </ol>
        </div>
    </asp:View>
    <asp:View ID="View2" runat="server">
        <ektronUI:Message ID="Message1" DisplayMode="Information" Visible="false"
runat="server" />
        <asp:ListView ID="ListView1" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
            Visible="false">
            <EmptyDataTemplate>
                Search criteria did not match any records.
            </EmptyDataTemplate>
            <LayoutTemplate>
                <table class="devsite-api-method">
                    <thead>
                        <tr>
                            <th>
                                Content Id
                            </th>
                            <th>
                                Content Title
                            </th>
                            <th>
                                Quicklink
                            </th>
                        </tr>
                    </thead>
                    <tbody>
                        <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                    </tbody>
                </table>
            </LayoutTemplate>
            <ItemTemplate>
            <h1 > FOR C# ASPX <h1 />
                <tr>
                    <td class="devsite-method" style="width: 55px">
                        <%#((SearchResultData)Container.DataItem)
[SearchContentProperty.Id].ToString()%>
                    </td>
                    <td class="devsite-method">
                        <%# ((SearchResultData)Container.DataItem)
[SearchContentProperty.Title].ToString()%>
```

```
                    </td>
                    <td class="devsite-method">
                        <%# ((SearchResultData)Container.DataItem)
[SearchContentProperty.QuickLink].ToString()%>
                    </td>
                </tr>
                <h1 > FOR VB ASPX <h1 />
                    <tr>
                        <td class="devsite-method" style="width: 55px">
                            <%# Container.DataItem(SearchContentProperty.Id).ToString
()%>
                        </td>
                        <td class="devsite-method">
                            <%# Container.DataItem(SearchContentProperty.Title).ToString
()%>
                        </td>
                        <td class="devsite-method">
                            <%# Container.DataItem
(SearchContentProperty.QuickLink).ToString()%>
                        </td>
                    </tr>
            </ItemTemplate>
        </asp:ListView>
    </asp:View>
</asp:MultiView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Search;
using Ektron.Cms.Search.Expressions;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            if (!String.IsNullOrWhiteSpace(uxSearchText.Text))
            {
                KeywordSearchCriteria criteria = new KeywordSearchCriteria();
                criteria.OrderBy = new List<OrderData>()
                {
                    new OrderData(SearchContentProperty.Rank, OrderDirection.Descending)
                };
                criteria.PagingInfo = new PagingInfo(10);
                criteria.PagingInfo.CurrentPage = 1;
                criteria.ReturnProperties = new HashSet<PropertyExpression>()
                {
                    SearchContentProperty.Id,
                    SearchContentProperty.Title,
```

```
                    SearchContentProperty.QuickLink
                };

                ContentManager contentManager = new ContentManager();

                Expression expressionTree =
Ektron.Cms.Search.SearchType.IsNonUserContent();

                if (contentManager.RequestInformation.ContentLanguage == 0)
                {
                    expressionTree &= SearchContentProperty.Language.EqualTo
(contentManager.RequestInformation.DefaultContentLanguage);
                }
                else if (contentManager.RequestInformation.ContentLanguage > 0)
                {
                    expressionTree &= SearchContentProperty.Language.EqualTo
(contentManager.RequestInformation.ContentLanguage);
                }

                criteria.ExpressionTree = expressionTree;

                //Pass in query text
                criteria.QueryText = uxSearchText.Text;

                ISearchManager manager = ObjectFactory.GetSearchManager();

                SearchResponseData response = manager.Search(criteria);
                uxSearchResultView.Visible = true;
                uxSearchResultView.DataSource = response.Results;
                uxSearchResultView.DataBind();

                uxPageMultiView.SetActiveView(uxViewMessage);
                if (response.PagingInfo.TotalRecords > 0)
                {
                    string pagingInfo = "Total # of Results = " +
response.PagingInfo.TotalRecords.ToString() + "<br />";
                    pagingInfo += "Total # of Pages = " +
response.PagingInfo.TotalPages.ToString() + "<br />";
                    pagingInfo += "Displaying search result " +
response.PagingInfo.StartRow.ToString() + " through " +
response.PagingInfo.EndRow.ToString();
                    MessageUtilities.UpdateMessage(uxMessage, pagingInfo,
Message.DisplayModes.Success);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter a search text",
Message.DisplayModes.Error);
                uxPageMultiView.SetActiveView(uxViewMessage);
            }
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

QueryPropositionManager

9.00 and higher

The QueryPropositionManager provides access to query the index of a configured search provider.

Namespace

```
Ektron.Cms.Framework.Search.QueryPropositionManager
```

Constructors

- `QueryPropositionManager()`
- `QueryPropositionManager(ApiAccessMode)`

Properties

- `QueryPropositionService`. Gets the underlying query proposition service supporting the configured search provider.

Methods

- *GetQueryCompletions* below
- `GetQueryCompletions(String,Int32,QueryCompletionSortOrder)`. Gets query completions as directed by the specified query completion request parameters.
- `GetQueryCompletions(QueryCompletionRequest)`. Gets query completions as directed by the specified query completion request parameters.
- *GetQuerySuggestions* on page 1344
- `GetQuerySuggestions(QuerySuggestionRequest)`. Return Query suggestions based on request parameters.

GetQueryCompletions

```
GetQueryCompletions(System.String,System.Int32)
```

Return query suggestions based on request parameters.

Parameters

- `searchText`. Search term
- `maxCount`. Suggested maximum count of query completions for the search engine. Actual counts may vary according to term availability as determined by the search engines internal algorithms.

Returns

Query completions.

.aspx code snippet

```
<div>
    <ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxSearchTextLabel" AssociatedControlID="uxSearchText"
CssClass="span-4 last"
                runat="server" Text="Search Text:" />
            <ektronUI:Autocomplete ID="uxSearchText" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
        </li>
        <li class="clearfix">
            <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Search"></ektronUI:Button>
        </li>
    </ol>
</div>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;
using Ektron.Cms.Search;
using Ektron.Cms.Search.Expressions;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            if (!String.IsNullOrWhiteSpace(uxSearchText.Text))
            {
                KeywordSearchCriteria criteria = new KeywordSearchCriteria();
                criteria.OrderBy = new List<OrderData>()
                {
                    new OrderData(SearchContentProperty.Rank, OrderDirection.Descending)
                };
                criteria.PagingInfo = new PagingInfo(10);
                criteria.PagingInfo.CurrentPage = 1;
                criteria.ReturnProperties = new HashSet<PropertyExpression>()
                {
                    SearchContentProperty.Id,
                    SearchContentProperty.Title,
                    SearchContentProperty.QuickLink
                };
```

```
                ContentManager contentManager = new ContentManager();

                Expression expressionTree =
Ektron.Cms.Search.SearchType.IsNonUserContent();

                if (contentManager.RequestInformation.ContentLanguage == 0)
                {
                    expressionTree &= SearchContentProperty.Language.EqualTo
(contentManager.RequestInformation.DefaultContentLanguage);
                }
                else if (contentManager.RequestInformation.ContentLanguage > 0)
                {
                    expressionTree &= SearchContentProperty.Language.EqualTo
(contentManager.RequestInformation.ContentLanguage);
                }

                criteria.ExpressionTree = expressionTree;

                //Pass in query text
                criteria.QueryText = uxSearchText.Text;

                ISearchManager manager = ObjectFactory.GetSearchManager();

                SearchResponseData response = manager.Search(criteria);
                uxSearchResultView.Visible = true;
                uxSearchResultView.DataSource = response.Results;
                uxSearchResultView.DataBind();

                uxPageMultiView.SetActiveView(uxViewMessage);
                if (response.PagingInfo.TotalRecords > 0)
                {
                    string pagingInfo = "Total # of Results = " +
response.PagingInfo.TotalRecords.ToString() + "<br />";
                    pagingInfo += "Total # of Pages = " +
response.PagingInfo.TotalPages.ToString() + "<br />";
                    pagingInfo += "Displaying search result " +
response.PagingInfo.StartRow.ToString() + " through " +
response.PagingInfo.EndRow.ToString();
                    MessageUtilities.UpdateMessage(uxMessage, pagingInfo,
Message.DisplayModes.Success);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter a search text",
Message.DisplayModes.Error);
                uxPageMultiView.SetActiveView(uxViewMessage);
            }
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetQuerySuggestions

```
GetQuerySuggestions(System.String,System.Int32)
```

Return Query suggestions based on request parameters.

Parameters

- `searchText`. Query text.
- `maxCount`. Maximum number of query completions requested.

Returns

Query suggestions.

Remarks

- `All Providers`. The terms returned for any request are subject to the internal algorithms of the configured search provider, the quality of the content in the index, and the configuration of any supporting dictionaries.

  The maximum count parameter is a suggestion for the search engine. Actual counts may vary according to term availability, as determined by the search engine's internal algorithms.

.aspx code snippet

```
<div>
    <ol class="formFields">
        <li class="clearfix">
            <ektronUI:Label ID="uxSearchTextLabel" AssociatedControlID="uxSearchText"
CssClass="span-4 last"
                runat="server" Text="Search Text:" />
            <ektronUI:Autocomplete ID="uxSearchText" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
        </li>
        <li class="clearfix">
            <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Search"></ektronUI:Button>
        </li>
    </ol>
</div>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;
using Ektron.Cms.Search;
using Ektron.Cms.Search.Expressions;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            if (!String.IsNullOrWhiteSpace(uxSearchText.Text))
            {
                KeywordSearchCriteria criteria = new KeywordSearchCriteria();
                criteria.OrderBy = new List<OrderData>()
                {
                    new OrderData(SearchContentProperty.Rank, OrderDirection.Descending)
                };
                criteria.PagingInfo = new PagingInfo(10);
                criteria.PagingInfo.CurrentPage = 1;
                criteria.ReturnProperties = new HashSet<PropertyExpression>()
                {
                    SearchContentProperty.Id,
                    SearchContentProperty.Title,
                    SearchContentProperty.QuickLink
                };

                 ContentManager contentManager = new ContentManager();

                 Expression expressionTree =
Ektron.Cms.Search.SearchType.IsNonUserContent();

                 if (contentManager.RequestInformation.ContentLanguage == 0)
                 {
                     expressionTree &= SearchContentProperty.Language.EqualTo
(contentManager.RequestInformation.DefaultContentLanguage);
                 }
                 else if (contentManager.RequestInformation.ContentLanguage > 0)
                 {
                     expressionTree &= SearchContentProperty.Language.EqualTo
(contentManager.RequestInformation.ContentLanguage);
                 }

                criteria.ExpressionTree = expressionTree;

                //Pass in query text
                criteria.QueryText = uxSearchText.Text;

                ISearchManager manager = ObjectFactory.GetSearchManager();

                SearchResponseData response = manager.Search(criteria);
                uxSearchResultView.Visible = true;
                uxSearchResultView.DataSource = response.Results;
                uxSearchResultView.DataBind();

                uxPageMultiView.SetActiveView(uxViewMessage);
                if (response.PagingInfo.TotalRecords > 0)
                {
                    string pagingInfo = "Total # of Results = " +
response.PagingInfo.TotalRecords.ToString() + "<br />";
                    pagingInfo += "Total # of Pages = " +
```

```
response.PagingInfo.TotalPages.ToString() + "<br />";
                    pagingInfo += "Displaying search result " +
response.PagingInfo.StartRow.ToString() + " through " +
response.PagingInfo.EndRow.ToString();
                    MessageUtilities.UpdateMessage(uxMessage, pagingInfo,
Message.DisplayModes.Success);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter a search text",
Message.DisplayModes.Error);
                uxPageMultiView.SetActiveView(uxViewMessage);
            }
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Search Data Classes

8.60 and higher

AdministratorPermission

The AdministratorPermission class serves as a token identifying that a search query is intended to be executed as an administrator user.

Namespace

```
Ektron.Cms.Search.AdministratorPermission
```

Methods

- `Authenticate(IAuthenticationHandler)`. Authenticates the current permission against the given authentication handler

AdvancedSearchCriteria

8.50 and higher

The AdvancedSearchCriteria class describes a narrow search query primarily defined by a logical expression tree. This criteria is intended narrower, targeted, querying.

Namespace

```
Ektron.Cms.Search.AdvancedSearchCriteria
```

Properties

- `EnableStemming`. Gets or sets a flag indicating whether or not stemming is enabled for the query.
- `ExpressionTree`. Gets and sets the logical expression tree for filtering results.
- `IncludeSuggestedResults`. Gets and sets a flag indicating whether to retrieve suggested results in addition to the relevant search results.
- `Locale`. Gets and sets the query engine's target locale.
- `OrderBy`. Gets and sets the list of result ordering rules.
- `PagingInfo`. Gets and sets the paging information.
- `Permission`. Gets and sets the permissions with which to search.
- `Refinement`. Gets or sets the criteria used to indicate how refinements should be applied to the query.

  NOTE: Query refinement is not supported by all search providers and may require some additional manual configuration.

- `ReturnProperties`. Gets and sets the list of columns to return in the results.
- `Scope`. Gets and sets the scope information.
- `Similarity`. Gets or sets the criteria used to refine the query to items that are most similar to a particular item. Note: Similarity search functionality is not supported by all search providers.

Constructors

- `AdvancedSearchCriteria()`. Constructor.

Remarks

- `All Providers`. The `Locale` property should be used to specify what type of language processing is applied to your query. The current site language is used as a default if the `Locale` is not explicitly specified.
- `FAST Search for Sharepoint 2010`. The `AdvancedSearchCriteria` is not applicable when your search provider is Microsoft FAST Search for SharePoint. For the purposes of easing migration to this provider, any queries using this criteria is interpreted as `KeywordSearchCriteria`. See also `KeywordSearchCriteria` for additional information.
- `Search Server 2010`. The `Refinement` property is not supported if your search provider is Microsoft Search Server 2010. Deep refinement of result sets is not available.

  The `Similarity` property is not supported if your search provider is Microsoft Search Server 2010. Similarity search functionality is not available.

- `Solr`. The `AdvancedSearchCriteria` is not applicable when your search provider is Apache Solr. For the purposes of easing migration to this provider, any queries using this criteria are interpreted as `KeywordSearchCriteria`. See also the `KeywordSearchCriteria` for additional information.

8.60 and higher

BoundedFacetBucket

The BoundedFacetBucket class is a data structure describing a facet result that represent a range of values.

Namespace

```
Ektron.Cms.Search.BoundedFacetBucket<T>
```

Properties

- `Count`. Gets or sets the count of items encompassed by this bucket.
- `Refinement`. Gets or sets the refinement data identifying this facet. This data can be applied to subsequent queries to indicate that the result set should be restricted to items classified by this FacetBucket.
- `Value`. Gets or sets the upper and lower bounds for data encompassed by this facet bucket.

Constructors

- `BoundedFacetBucket(BoundedValue<T>,Int64)`. Constructor.

8.60 and higher

BoundedValue

The BoundedValue class is a data structure representing a range of refinement values.

Namespace

```
Ektron.Cms.Search.BoundedValue<U>
```

Properties

- `LowerBound`. Gets or sets the lower bound of the value range.
- `UpperBound`. Gets or sets the upper bound of the value range.

Constructors

- `BoundedValue(U)`. Constructor.
- `BoundedValue(U,U)`. Constructor.

CurrentUserPermission

8.50 and higher

The CurrentUserPermission class serves as a token identifying that a search query is intended to be executed as the user that has triggered the action to issue it.

Namespace

```
Ektron.Cms.Search.CurrentUserPermission
```

Methods

- `Authenticate(IAuthenticationHandler)`. Authenticates the current permission against the given authentication handler.

DateFacet

8.60 and higher

The DateFacet class represents a classification of search results on a date/time index field.

Namespace

```
Ektron.Cms.Search.DateFacet
```

Properties

- `Buckets`. Gets or sets the bounded facet buckets representing the DateTime ranges for this facet.
- `Property`. Gets or sets the property associated with this facet.

Constructors

- `DateFacet(DatePropertyExpression)`. Constructor.
- `DateFacet (DatePropertyExpression,ICollection<BoundedFacetBucket<DateTime>>)`. Constructor.
- `DateFacet(DateMultiValuePropertyExpression)`. Constructor.
- `DateFacet (DateMultiValuePropertyExpression,ICollection<BoundedFacetBucket<DateTime>>)`. Constructor.
- `DateFacet(PropertyExpression)`. Constructor.
- `DateFacet (PropertyExpression,ICollection<BoundedFacetBucket<DateTime>>)`. Constructor.

DateRefinementSpecification

8.60 and higher

The DateRefinementSpecification represents a strongly typed refinement specification for date/time index properties.

Namespace

```
Ektron.Cms.Search.DateRefinementSpecification
```

Properties

- `Property`. Gets or sets the refinement property.

Constructors

- `DateRefinementSpecification()`. Constructor.
- `DateRefinementSpecification(DateMultiValuePropertyExpression)`. Constructor.
- `DateRefinementSpecification (DateMultiValuePropertyExpression,ICollection<BoundedValue<DateTime>>)`. Constructor.
- `DateRefinementSpecification(DatePropertyExpression)`. Constructor.
- `DateRefinementSpecification (DatePropertyExpression,ICollection<BoundedValue<DateTime>>)`. Constructor.

Remarks

- `All Providers`. Add a `DateRefinementSpecification` to search criteria to identify a field for which facet data should be returned with the query results.
- `FAST Search for Sharepoint 2010`. Specifying explicit bounds for date refinements is not supported when your search provider is Microsoft FAST Search for SharePoint 2010. The date ranges are auto-generated by the search engine.
- `Search Server 2010`. The `DateRefinementSpecification` is not applicable if your search provider is Microsoft Search Server 2010. Deep refinement of result sets is not available.
- `Solr`. When explicit bounds for date refinements are not specified each unique date value will represent an individual facet bucket in your response.

DecimalFacet

8.60 and higher

The DecimalFacet class represents a classification of search results on a decimal index field.

Namespace

```
Ektron.Cms.Search.DecimalFacet
```

Properties

- `Buckets`. Gets or sets the collection of facet buckets associated with this facet.
- `Property`. Gets or sets the property associated with this facet.

Constructors

- `DecimalFacet(DecimalMultiValuePropertyExpression)`. Constructor.
- `DecimalFacet (DecimalMultiValuePropertyExpression,ICollection<BoundedFacetBucket<Double>>)`. Constructor.
- `DecimalFacet(DecimalPropertyExpression)`. Constructor.
- `DecimalFacet (DecimalPropertyExpression,ICollection<BoundedFacetBucket<Double>>)`. Constructor.
- `DecimalFacet(PropertyExpression)`. Constructor.
- `DecimalFacet (PropertyExpression,ICollection<BoundedFacetBucket<Double>>)`. Constructor.

8.60 and higher

DecimalRefinementSpecification

The DecimalRefinementSpecification represents a strongly typed refinement specification for decimal index properties.

Namespace

```
Ektron.Cms.Search.DecimalRefinementSpecification
```

Properties

- `Property`. Gets or sets the refinement property.

Constructors

- `DecimalRefinementSpecification()`. Constructor.
- `DecimalRefinementSpecification (DecimalMultiValuePropertyExpression)`. Constructor.
- `DecimalRefinementSpecification (DecimalMultiValuePropertyExpression,ICollection<BoundedValue<Double>>)`. Constructor.
- `DecimalRefinementSpecification(DecimalPropertyExpression)`. Constructor.

- `DecimalRefinementSpecification (DecimalPropertyExpression,ICollection<BoundedValue<Double>>)`. Constructor.

Remarks

- `All Providers`. Add a `DecimalRefinementSpecification` to search criteria to identify a field for which facet data should be returned with the query results.
- `FAST Search for Sharepoint 2010`. Specifying explicit bounds for decimal refinements is not supported when your search provider is Microsoft FAST Search for SharePoint 2010. The decimal ranges are auto-generated by the search engine.
- `Search Server 2010`. The `DecimalRefinementSpecification` is not applicable if your search provider is Microsoft Search Server 2010. Deep refinement of result sets is not available.
- `Solr`. When explicit bounds for decimal refinements are not specified each unique decimal value will represent an individual facet bucket in your response.

Facet

8.60 and higher

The Facet class represents a classification of search results on a specific index field.

Namespace

```
Ektron.Cms.Search.Facet<T>
```

Properties

- `Property`. Gets or sets the property associated with this facet.

Constructors

- `Facet(T)`. Constructor.

FacetBucket

8.60 and higher

The FacetBucket class describes a subset of results classified within a given facet.

Namespace

```
Ektron.Cms.Search.FacetBucket
```

Properties

- `Count`. Gets or sets the count of items encompassed by this bucket.
- `Refinement`. Gets or sets the refinement data identifying this facet. This data can be applied to subsequent queries to indicate that the result set should be restricted to items classified by this FacetBucket.

Constructors

- `FacetBucket(Int64)`. Constructor.

Facets

8.60 and higher

The Facets class is a data structure encapsulating a collection of facet data for an individual search response.

Namespace

```
Ektron.Cms.Search.Facets
```

Properties

- `Item[DateMultiValuePropertyExpression]`. Gets a DateFacet associated with the specified property.
- `Item[DatePropertyExpression]`. Gets a DateFacet associated with the specified property.
- `Item[DecimalMultiValuePropertyExpression]`. Gets a DecimalFacet associated with the specified property.
- `Item[DecimalPropertyExpression]`. Gets a DecimalFacet associated with the specified property.
- `Item[IntegerMultiValuePropertyExpression]`. Gets a IntegerFacet associated with the specified property.
- `Item[IntegerPropertyExpression]`. Gets a IntegerFacet associated with the specified property.
- `Item[PropertyExpression]`. Gets a Facet associated with the specified property.
- `Item[String]`. Gets a Facet with the specified property name.
- `Item[StringMultiValuePropertyExpression]`. Gets a StringFacet associated with the specified property.
- `Item[StringPropertyExpression]`. Gets a StringFacet associated with the specified property.

Constructors

- `Facets()`. Constructor.
- `Facets(ICollection<Facet<PropertyExpression>>)`. Constructor.

Methods

- `HasFacet(PropertyExpression)`. Returns true if this instance contains a facet for the property represented by the specified PropertyExpression.
- `HasFacet(String)`. Returns true if this instance contains a facet for the property represented by the specified property name.

IntegerFacet

8.60 and higher

The IntegerFacet class represents a classification of search results on a integer index field.

Namespace

```
Ektron.Cms.Search.IntegerFacet
```

Properties

- `Buckets`. Gets or sets the collection of facet buckets associated with this facet.
- `Property`. Gets or sets the property associated with this facet.

Constructors

- `IntegerFacet(IntegerMultiValuePropertyExpression)`. Constructor.
- `IntegerFacet (IntegerMultiValuePropertyExpression,ICollection<BoundedFacetBucket<Int64>>)`. Constructor.
- `IntegerFacet(IntegerPropertyExpression)`. Constructor.
- `IntegerFacet (IntegerPropertyExpression,ICollection<BoundedFacetBucket<Int64>>)`. Constructor.
- `IntegerFacet(PropertyExpression)`. Constructor.
- `IntegerFacet (PropertyExpression,ICollection<BoundedFacetBucket<Int64>>)`. Constructor.

IntegerRefinementSpecification

8.60 and higher

The IntegerRefinementSpecification represents a strongly typed refinement specification for integer index properties.

Namespace

```
Ektron.Cms.Search.IntegerRefinementSpecification
```

Properties

- `Property`. Gets or sets the refinement property.

Constructors

- `IntegerRefinementSpecification()`. Constructor.
- `IntegerRefinementSpecification (IntegerMultiValuePropertyExpression)`. Constructor.
- `IntegerRefinementSpecification (IntegerMultiValuePropertyExpression,ICollection<BoundedValue<Int64>>)`. Constructor.
- `IntegerRefinementSpecification(IntegerPropertyExpression)`. Constructor.
- `IntegerRefinementSpecification (IntegerPropertyExpression,ICollection<BoundedValue<Int64>>)`. Constructor.

Remarks

- `All Providers`. Add a `IntegerRefinementSpecification` to search criteria to identify a field for which facet data should be returned with the query results.
- `FAST Search for Sharepoint 2010`. Specifying explicit bounds for integer refinements is not supported when your search provider is Microsoft FAST Search for SharePoint 2010. The integer ranges are auto-generated by the search engine.
- `Search Server 2010`. The `IntegerRefinementSpecification` is not applicable if your search provider is Microsoft Search Server 2010. Deep refinement of result sets is not available.
- `Solr`. When explicit bounds for integer refinements are not specified each unique integer value will represent an individual facet bucket in your response.

8.50 and higher

KeywordSearchCriteria

The KeywordSearchCriteria class describes a free-text, keyword-centric search query. The criteria is intended for broad, traditional search queries.

Namespace

```
Ektron.Cms.Search.KeywordSearchCriteria
```

Properties

- `EnableStemming`. Gets or sets a flag indicating whether or not stemming is enabled for the query.
- `ExpressionTree`. Gets and sets the logical expression tree for filtering results.
- `ImplicitAnd`. Gets or sets a flag indicating whether or not an 'AND' operator is implied between search terms that lack any other explicit logical operator.
- `IncludeSuggestedResults`. Gets and sets a flag indicating whether or not to retrieve suggested results in addition to the relevant search results.
- `Locale`. Gets and sets the query engine's target locale.
- `OrderBy`. Gets and sets the list of result ordering rules.
- `PagingInfo`. Gets and sets the paging information.
- `Permission`. Gets and sets the permissions with which to search.
- `QueryText`. Gets and sets the keyword query text with which to filter the search results.
- `Refinement`. Gets or sets the criteria used to indicate how refinements should be applied to the query.

  NOTE: Query refinement is not supported by all search providers and may require some additional manual configuration.

- `ReturnProperties`. Gets and sets the list of columns to return in the results.
- `Scope`. Gets and sets the scope information.
- `Similarity`. Gets or sets the criteria used to refine the query to items that are most similar to a particular item. Note: Similarity search functionality is not supported by all search providers.

Constructors

- `KeywordSearchCriteria()`. Constructor.

Remarks

- `All Providers`. As a best practice, explicit property names should not be hard-coded in the value of the `QueryText` property. The PropertyExpression class (and its various derived classes) exist to abstract away the resolution of property names. This serves to decouple your query from the underlying index schema as well as maximize portability and version compatibility.

  The `Locale` property should be used to specify what type of language processing is applied to your query. The current site language is used as a default if the Locale is not explicitly specified.

- `FAST Search for Sharepoint 2010`. The Permission property is not supported if your search provider is Microsoft FAST Search for SharePoint 2010. Security trimming of documents at query time is not available.
- `Search Server 2010`. The `Refinement` property is not supported if your search provider is Microsoft Search Server 2010. Deep refinement of result sets is not available.

  The `Similarity` property is not supported if your search provider is Microsoft Search Server 2010. Similarity search functionality is not available.
- `Solr`. With Apache Solr, the `EnableStemming` option is not available. In Solr, stemming is not a feature that can be toggled at query-time. It is configured on a field by field basis in the `schema.xml` for a given Solr core. By default, stemming is only applied to the content body and content title. There is no stemming applied to any of the other index fields. If a different behavior is required, you should consider customizing the index schema for your site.

  The `Permission` property is not supported if your search provider is Apache Solr. Security trimming of documents at query time is not available.

  The `Similarity` property is not supported if your search provider is Apache Solr. Similarity search functionality is not available.

  The `Scope` property is not supported if your search provider is Apache Solr.

  Ranking algorithms will use the contents of the `QueryText` property when determining the relevancy of a particular result item. The `ExpressionTree` will filter your results but will not influence their rank.

  While Solr utilizes boolean operators and related terminology, it is important to consider that these expressions are actually interpreted by Solr as set operations. The 'AND' expression can be thought of as a conjunction operation and 'OR' can be thought of as an injunction operation. If your query produces an unexpected result set, try to revise your query with this in mind. (For additional details on this behavior, see the following article: Why Not AND, OR, And NOT?)

  By default, the content of assets (Word documents, PDFs, and so on) is indexed for files up to 10 MB in size. This limit is intended to accelerate crawl times and minimize the burden placed upon the search server by the content extraction process. When a file exceeds this limit, its summary is indexed instead of the full content of the document. The limit can be adjusted via the ManifoldCF administration console. (Use caution when updating advanced configuration options such as this. Changes may impact the performance of your search server.)

  Solr relies on Apache Tika, an open source collection of content analysis libraries, for the extraction of content from asset files (Word documents, PDFs, and so on). Support for the extraction of content from these types of files is subject to the features and limitations of this toolkit.

Locale

8.50 and higher

The Locale class provides a mapping of language IDs to the relevant locale for a search query.

Namespace

```
Ektron.Cms.Search.Locale
```

Properties

- `Code`. Gets the identifying code for this locale.
- `Default`. Gets the default 'Locale' according to the current content language.
- `ID`. Gets the language ID for this locale.

Methods

- `Create(Int32)`. Gets the string locale code represented by the language ID.
- `Get(Int32)`. Gets the locale code for the specified language ID.

ManualUserPermission

8.50 and higher

The ManualUserPermission class is a data structure encapsulating permission data representing a specific, manually specified, CMS user.

Namespace

```
Ektron.Cms.Search.ManualUserPermission
```

Properties

- `UserId`. Gets and sets the user identification to apply to search results

Constructors

- `ManualUserPermission(Int64)`. Constructor.

Methods

- `Authenticate(IAuthenticationHandler)`. Authenticates the current permission against the given authentication handler.

OrderData

8.50 and higher

The OrderData class represents the ordering information for the result set of a search query.

Namespace

```
Ektron.Cms.Search.OrderData
```

Properties

- `Direction`. Gets or sets the direction in which to sort a result set.
- `Property`. Gets or sets the property on which to sort a result set.

Constructors

- `OrderData(PropertyExpression,OrderDirection)`. Constructor.

Remarks

- `All Providers`. Your search provider may require that you explicitly configure an index field as sortable. If you find that your results are not sorting as directed, consult the search provider vendor's documentation for information regarding how you would configure this functionality.

  Many search providers will apply an ASCII-based sort to textual fields. Keep this in mind when sorting on string properties.

  The `HighlightedSummary` property cannot be sorted.

  The `Rank` property cannot be sorted in an ascending direction.

- `Solr`. Sorting of results is not supported for multi-value fields.

  Sorting of results is not supported for text fields with tokenization or other text processing applied to them. If you are sorting on a custom field, it is recommended that you also create an exact-match companion field containing an unprocessed copy of the data. See your `schema.xml` file for additional information on creating these fields.

  Smart Form fields cannot be sorted on. Apache Solr does not support the sorting of multi-value fields. If sorting behavior is required for data associated with Smart Form content, please consider using a companion metadata definition.

Permission

8.50 and higher

The Permission class is an abstract base class describing CMS permission data that can be applied to a search query.

Namespace

```
Ektron.Cms.Search.Permission
```

Methods

- `Authenticate(IAuthenticationHandler)`. Authenticates the current permission against the given authentication handler.
- `CreateAdministratorPermission()`. Creates a new permission instance representing a CMS administrator.
- `CreateCurrentUserPermission()`. Creates a new Permission instance representing the current CMS user.
- `CreateManualUserPermission(Int64)`. Creates a new Permission instance representing the specified CMS user.

9.00 and higher

QueryCompletionRequest

The QueryCompletionRequest class defines the parameters of a request for the configured search engine to predict query terms based on some partial term input.

Namespace

```
Ektron.Cms.Search.QueryCompletionRequest
```

Properties

- `DictionaryName`. Gets or sets the name of the target dictionary supplying completed terms. This parameter is not supported by all search providers. See the Ektron Developer Reference for guidance.
- `HandlerName`. Gets or sets the name of the handler supplying completed terms. This parameter is not supported by all search providers. See the Ektron Developer Reference for guidance.
- `Language`. Gets or sets the target content language for content feeding potential completed terms.
- `MaxCount`. Gets or sets the suggested maximum count of query completions for the search engine. Actual counts may vary according to term availability as determined by the search engine's internal algorithms.
- `Query`. Gets or sets the (partial) query term which the search engine will use as a seed for predicting completed terms.
- `SortOrder`. Gets or sets the type of sorting applied to query completion results.

Constructors

- `QueryCompletionRequest()`. Constructor.
- `QueryCompletionRequest(String,Int32)`. Constructor.

- `QueryCompletionRequest (String,Int32,Int32,QueryCompletionSortOrder)`. Constructor.
- `QueryCompletionRequest(String,Int32,QueryCompletionSortOrder)`. Constructor.

Remarks

- `All Providers`. The maximum count parameter is a suggestion for the search engine. Actual counts may vary according to term availability, as determined by the search engine's internal algorithms.
- `FAST Search for Sharepoint 2010`. The `Language` property is not supported if your search provider is Microsoft FAST Search for SharePoint 2010.

  The `HandlerName` property is not supported if your search provider is Microsoft FAST Search for SharePoint 2010.

  The `DictionaryName` property is not supported if your search provider is Microsoft FAST Search for SharePoint 2010.
- `Search Server 2010`. The `Language` property is not supported if your search provider is Microsoft Search Server 2010.

  The `HandlerName` property is not supported if your search provider is Microsoft Search Server 2010.

  The `DictionaryName` property is not supported if your search provider is Microsoft Search Server 2010.

QueryCompletionResponse

9.00 and higher

The QueryCompletionResponse class is a data structure which aggregates query completions for a partial search term, as predicted by the configured search provider.

Namespace

```
Ektron.Cms.Search.QueryCompletionResponse
```

Properties

- `QueryCompletions`. Gets or sets a collection of completed terms as predicted by the configured search provider.

Constructors

- `QueryCompletionResponse()`. Constructor.

QueryCompletionTerm

9.00 and higher

The QuerySuggestionTerm class is a data structure describing an individual, completed, query term as predicted by the configured search provider.

Namespace

```
Ektron.Cms.Search.QuerySuggestionTerm
```

Properties

- `Suggestion`. Gets or sets the predicted query term.

Constructors

- `QuerySuggestionTerm()`. Constructor.

QuerySuggestionResponse

9.00 and higher

The QuerySuggestionResponse class is a data structure which aggregates query completions for a partial search term, as predicted by the configured search provider.

Namespace

```
Ektron.Cms.Search.QuerySuggestionResponse
```

Properties

- `QuerySuggestions`. Gets or sets a collection of suggested terms as predicted by the configured search provider.

Constructors

- `QuerySuggestionResponse()`. Constructor.

QuerySuggestionTerm

9.00 and higher

The QuerySuggestionTerm class is a data structure describing an individual, completed, query term as predicted by the configured search provider.

Namespace

```
Ektron.Cms.Search.QuerySuggestionTerm
```

Properties

- `Suggestion`. Gets or sets the predicted query term.

Constructors

- `QuerySuggestionTerm()`. Constructor.

8.60 and higher

Refinement

The Refinement class is a data structure identifying a specific subset of results to which a query should be restricted.

Namespace

```
Ektron.Cms.Search.Refinement
```

Properties

- `Data`. Gets or sets the data identifying this refinement.
- `Property`. Gets or sets the property associated with this refinement.

Constructors

- `Refinement()`. Constructor.
- `Refinement(PropertyExpression,String)`. Constructor.

Remarks

- `All Providers`. Add a `Refinement` to your search criteria to restrict a query to results classified by a specific facet bucket. Every `FacetBucket` includes a reference to a `Refinement`, which can be added to a subsequent query.
- `Search Server 2010`. The `Refinement` is not applicable if your search provider is Microsoft Search Server 2010. Deep refinement of result sets is not available.

RefinementCriteria

8.60 and higher

The RefinementCriteria class encapsulates all query criteria related to faceted search.

Namespace

```
Ektron.Cms.Search.RefinementCriteria
```

Properties

- `IsEnabled`. Gets or sets a flag indicating whether or not refinement functionality is enabled for a query.
- `Items`. Gets a read-only collection of the refinement information associated with this criteria.

Constructors

- `RefinementCriteria()`. Constructor.

Methods

- `Add(Refinement)`. Adds a specific refinement identifying a specific subset of results to which the query should be restricted.
- `Add(RefinementInfo)`. Add a `RefinementInfo` data structure to the criteria.
- `Add(RefinementSpecification<PropertyExpression>)`. Adds a RefinementSpecification identifying the facets that should be generated at query time.
- `Clear()`. Clears this criteria of any refinement specifications or restrictions that it may contain.

Remarks

- `All Providers`. Add a `RefinementSpecification` to this criteria to identify a field for which facet data should be returned with the query results.

  Add a `Refinement` to this criteria to restrict a query to results classified by a specific facet bucket. Every `FacetBucket` includes a reference to a `Refinement`, which can be added to a subsequent query.
- `Search Server 2010`. `RefinementCriteria` is not applicable if your search provider is Microsoft Search Server 2010. Deep refinement of result sets is not available.

RefinementInfo

8.60 and higher

The RefinementInfo class is a data structure describing individual refinement specifications and restrictions applied to a query.

Namespace

```
Ektron.Cms.Search.RefinementInfo
```

Properties

- `Property`. Gets or sets the property associated with this refinement information.
- `Refinements`. Gets the query refinement restrictions.
- `Specification`. Get or sets the query refinement specification.

Constructors

- `RefinementInfo()`. Constructor.
- `RefinementInfo(List<Refinement>)`. Constructor.
- `RefinementInfo(Refinement)`. Constructor.
- `RefinementInfo(RefinementSpecification<PropertyExpression>)`. Constructor.

RefinementSpecification

8.60 and higher

The RefinementSpecification class identifies a specific facet to be included in the search results.

Namespace

```
Ektron.Cms.Search.RefinementSpecification<T>
```

Properties

- `Property`. Gets or sets the refinement property.

Constructors

- `RefinementSpecification(T)`. Constructor.

Remarks

- `All Providers`. Add a `RefinementSpecification` to search criteria to identify a field for which facet data should be returned with the query results.
- `Search Server 2010`. The `RefinementSpecification` is not applicable if your search provider is Microsoft Search Server 2010. Deep refinement of result sets is not available.

SearchContentProperty

8.50 and higher

The ContentProperty class provides access to the PropertyExpressions which map default content properties to index fields.

Namespace

```
Ektron.Cms.Search.SearchContentProperty
```

Properties

- `AssetVersion`. Gets the PropertyExpression for the asset version path field.
- `Author`. Gets the PropertyExpression for the content author field.
- `ContentSubType`. Gets the PropertyExpression for the content subtype field.
- `ContentType`. Gets the PropertyExpression for the content type field.
- `DateCreated`. Gets the PropertyExpression for the content date created field.
- `DateModified`. Gets the PropertyExpression for the content date modified field.
- `Description`. Gets the PropertyExpression for the content description field.
- `ExpiryDate`. Gets the PropertyExpression for the content expiry date field.
- `ExpiryType`. Gets the PropertyExpression for the content expiry type field.
- `FolderId`. Gets the PropertyExpression for the folder ID field.
- `FolderIdPath`. Gets the PropertyExpression for the content folder ID path field.
- `FolderName`. Gets the PropertyExpression for the content folder name field.
- `FolderPath`. Gets the PropertyExpression for the content folder path field.

  NOTE: For more information on the `Folder` properties, see Recursive Folder Queries.

- `GoLiveDate`. Gets the PropertyExpression for the content go live date field.
- `HighlightedSummary`. Gets the PropertyExpression for the highlighted summary field.
- `Id`. Gets the PropertyExpression for the content ID field.
- `Language`. Gets the PropertyExpression for the content language field.
- `MapAddress`. Gets the PropertyExpression for the map address field.
- `MapDate`. Gets the PropertyExpression for the map date field.
- `MapLatitude`. Gets the PropertyExpression for the map latitude field.
- `MapLongitude`. Gets the PropertyExpression for the map longitude field.
- `Path`. Gets the PropertyExpression for the content path field.
- `Private`. Gets the PropertyExpresion for the field indicating whether or not the content is private.
- `QuickLink`. Gets the PropertyExpression for the QuickLink field.
- `Rank`. Gets the PropertyExpression for the rank field.
- `SiteId`. Gets the PropertyExpression for the ID of the parent site.
- `Size`. Gets the PropertyExpression for the content size field.
- `Tags`. Gets the PropertyExpression for the content tags field.

- `TaxonomyCategory`. Gets the PropertyExpression for the taxonomy category field.
- `Title`. Gets the PropertyExpression for the content title field.
- `XmlConfigId`. Gets the PropertyExpression for the content SmartForm ID field.

Remarks

- `All Providers`. The `HighlightedSummary` property expression is used to request a dynamically generated summary for each result item in your result set. This behavior varies from search provider to search provider. Its effectiveness also varies based on the quantity and quality of the data available in the index for a given piece of content. For example, XML-based content, such as Smart Form data, is less likely to render a meaningful summary than traditional textual content. It is recommended that static summaries are conditionally displayed for result items in cases where the dynamically generated one is insufficient.
- `Solr`. The `TaxonomyCategory` property is a multi-value field in Solr. Multi-value fields provide a better development experience when consuming field data representing a collection of values. It is recommended that you reference the `TaxonomyCategory` property as a multi-value property expression to ensure that you are always working with the complete data for your result item (See: StringMultiValuePropertyExpression class):

  ```
  SearchContentProperty.TaxonomyCategory.AsMultiValue()
  ```

  General keyword searches of hierarchical field data are not supported. Fields such as `TaxonomyCategory`, `FolderPath`, and `FolderIdPath` have been optimized for queries that are constructed using hierarchical paths. When querying this type of data, your input should be a rooted, fully formed path. For example, consider a taxonomy category of "\Departments\HR\Benefits." A query, against the taxonomy category field, with an input of "\Departments\HR," would return results associated with the category. A query with an input of "HR" would not.

  Solr will analyze the first 100,000 characters of an individual document when generating a `HighlightedSummary` value. For large content, a dynamic summary will not be generated if the search terms for your query are not found in the first 100,000 characters of the document.

SearchCriteria

8.50 and higher

The SearchCriteria class serves as a base for criteria defining a particular search query.

Namespace

```
Ektron.Cms.Search.SearchCriteria
```

Properties

- `EnableStemming`. Gets or sets a flag indicating whether or not stemming is enabled for the query.
- `ExpressionTree`. Gets and sets the logical expression tree for filtering results
- `IncludeSuggestedResults`. Gets and sets a flag indicating whether or not to retrieve suggested results in addition to the relevant search results.
- `Locale`. Gets and sets the query engine's target locale.
- `OrderBy`. Gets and sets the list of result ordering rules
- `PagingInfo`. Gets and sets the paging information
- `Permission`. Gets and sets the permissions with which to search
- `Refinement`. Gets or sets the criteria used to indicate how refinements should be applied to the query.

  NOTE: Query refinement is not supported by all search providers and may require some additional manual configuration.

- `ReturnProperties`. Gets and sets the list of columns to return in the results
- `Scope`. Gets and sets the scope information
- `Similarity`. Gets or sets the criteria used to refine the query to items that are most similar to a particular item. Note: Similarity search functionality is not supported by all search providers.

Constructors

- `SearchCriteria()`. Constructor.

SearchCustomProperty

8.50 and higher

The SearchCustomProperty class is a factory for property expressions targeting custom property fields.

Namespace

```
Ektron.Cms.Search.SearchCustomProperty
```

Methods

- `GetBooleanProperty(String)`. Gets a `BooleanPropertyExpression` representing the custom property field identified by the specified name value.
- `GetDateProperty(String)`. Gets a `DatePropertyExpression` representing the custom property field identified by the specified name value.
- `GetDecimalProperty(String)`. Gets a `DecimalPropertyExpression` representing the custom property field identified by the specified name value.

- `GetIntegerProperty(String)`. Gets a `IntegerPropertyExpression` representing the custom property field identified by the specified name value.
- `GetStringProperty(String)`. Gets a `StringPropertyExpression` representing the custom property field identified by the specified name value.

SearchECommerceProperty

8.50 and higher

The SearchECommerceProperty class provides access to the default PropertyExpressions for formulating e-commerce queries.

Namespace

```
Ektron.Cms.Search.SearchECommerceProperty
```

Properties

- `Buyable`. Gets the PropertyExpression for the buyable field.
- `CatalogNumber`. Gets the PropertyExpression for product folder ID field.
- `CurrencyId`. Gets the PropertyExpression for the currency ID field.
- `Description`. Gets the PropertyExpression for the product description field.
- `Discontinued`. Gets the PropertyExpression for the product discontinued field.
- `Height`. Gets the PropertyExpression for the product height field.
- `HighlightedSummary`. Gets the PropertyExpression for the highlighted product summary field.
- `Id`. Gets the PropertyExpression for product content ID field.
- `ImageUrl`. Gets the PropertyExpression for the product image field.
- `Language`. Gets the PropertyExpression for the product content language field.
- `Length`. Gets the PropertyExpression for the product length field.
- `ListPrice`. Gets the PropertyExpression for the list price field.
- `PathCategory`. Gets the PropertyExpression for the product content path field.
- `ProductType`. Gets the PropertyExpression for the product type field.
- `ProductTypeId`. Gets the PropertyExpression for the product type ID field.
- `Purchased`. Gets the PropertyExpression for the purchased field.
- `QuickLink`. Gets the PropertyExpression for the product QuickLink field.
- `Rank`. Gets the PropertyExpression for the result rank field.
- `SalePrice`. Gets the PropertyExpression for the sale price field.
- `Size`. Gets the PropertyExpression for the product content size field.
- `SkuNumber`. Gets the PropertyExpression for the SKU number field.
- `Title`. Gets the PropertyExpression for the product title field.
- `Weight`. Gets the PropertyExpression for the product weight field.
- `Width`. Gets the PropertyExpression for the product width field.

Remarks

- `All Providers`. The `HighlightedSummary` property expression is used to request a dynamically generated summary for each result item in your result set. This behavior varies from search provider to search provider. Its effectiveness also varies based on the quantity and quality of the data available in the index for a given piece of content. For example, XML-based content, such as Smart Form data, is less likely to render a meaningful summary than traditional textual content. It is recommended that static summaries are conditionally displayed for result items in cases where the dynamically generated one is insufficient.
- `Solr`. The `TaxonomyCategory` property is a multi-value field in Solr. Multi-value fields provide a better development experience when consuming field data representing a collection of values. It is recommended that you reference the `TaxonomyCategory` property as a multi-value property expression to ensure that you are always working with the complete data for your result item (See: StringMultiValuePropertyExpression class):

  ```
  SearchContentProperty.TaxonomyCategory.AsMultiValue()
  ```

  General keyword searches of hierarchical field data is not supported. Fields such as `TaxonomyCategory`, `FolderPath`, and `FolderIdPath` have been optimized for queries that are programmatically constructed. When querying this type of data, your input should be a rooted, fully formed, path. For example, consider a taxonomy category of "\Departments\HR\Benefits". A query, against the taxonomy category field, with an input of "\Departments\HR" would return results associated with the category. A query with an input of "HR" would not.

  Solr will analyze the first 100,000 characters of an individual document when generating a `HighlightedSummary` value. For large content, a dynamic summary will not be generated if the search terms for your query are not found in the first 100,000 characters of the document.

SearchGroupProperty

8.50 and higher

The GroupProperty class provides access to the default PropertyExpressions for formulating community group queries.

Namespace

```
Ektron.Cms.Search.SearchGroupProperty
```

Properties

- `Avatar`. Gets the PropertyExpression for the user avatar field.
- `CreatorId`. Gets the PropertyExpression for the group creator ID field.
- `DateModified`. Gets the PropertyExpression for the date modified field.

- `EmailAddress`. Gets the PropertyExpression for the email address field.
- `Enroll`. Gets the PropertyExpression for the enroll field.
- `HighlightedSummary`. Gets the PropertyExpression for the highlighted summary field.
- `Id`. Gets the PropertyExpression for the group ID field.
- `Location`. Gets the PropertyExpression for the location field.
- `LongDescription`. Gets the PropertyExpression for the long description field.
- `Name`. Gets the PropertyExpression for the name field.
- `Path`. Gets the PropertyExpression for the path field.
- `QuickLink`. Gets the PropertyExpression for the QuickLink field.
- `Rank`. Gets the PropertyExpression for the rank field.
- `ShortDescription`. Gets the PropertyExpression for the short description field.
- `Size`. Gets the PropertyExpression for the size field.
- `Tags`. Gets the PropertyExpression for the tags field.
- `TaxonomyCategory`. Gets the PropertyExpression for the taxonomy category field.
- `Title`. Gets the PropertyExpression for the title field.
- `Type`. Gets the PropertyExpression for the user type field.

Remarks

- `All Providers`. The `HighlightedSummary` property expression is used to request a dynamically generated summary for each result item in your result set. This behavior varies from search provider to search provider. Its effectiveness also varies based on the quantity and quality of the data available in the index for a given piece of content. For example, XML-based content, such as Smart Form data, is less likely to render a meaningful summary than traditional textual content. It is recommended that static summaries are conditionally displayed for result items in cases where the dynamically generated one is insufficient.
- `Solr`. The `TaxonomyCategory` property is a multi-value field in Solr. Multi-value fields provide a better development experience when consuming field data representing a collection of values. It is recommended that you reference the `TaxonomyCategory` property as a multi-value property expression to ensure that you are always working with the complete data for your result item (See: StringMultiValuePropertyExpression class):

  ```
  SearchContentProperty.TaxonomyCategory.AsMultiValue()
  ```

  General keyword searches of hierarchical field data is not supported. Fields such as `TaxonomyCategory`, `FolderPath`, and `FolderIdPath` have been optimized for queries that are programmatically constructed. When querying this type of data, your input should be a rooted, fully formed, path. For example, consider a taxonomy category of "\Departments\HR\Benefits". A query, against the

taxonomy category field, with an input of "\Departments\HR" would return results associated with the category. A query with an input of "HR" would not.

Solr will analyze the first 100,000 characters of an individual document when generating a `HighlightedSummary` value. For large content, a dynamic summary will not be generated if the search terms for your query are not found in the first 100,000 characters of the document.

SearchMetadataProperty

8.50 and higher

The SearchMetadataProperty class is a factory for property expressions targeting metadata fields.

Namespace

```
Ektron.Cms.Search.SearchMetadataProperty
```

Methods

- `GetBooleanProperty(String)`. Gets a `BooleanPropertyExpression` representing the metadata field identified by the specified name value.
- `GetDateProperty(String)`. Gets a `DatePropertyExpression` representing the metadata field identified by the specified name value.
- `GetDecimalProperty(String)`. Gets a `DecimalPropertyExpression` representing the metadata field identified by the specified name value.
- `GetIntegerProperty(String)`. Gets a `IntegerPropertyExpression` representing the metadata field identified by the specified name value.
- `GetStringProperty(String)`. Gets a `StringPropertyExpression` representing the metadata field identified by the specified name value.

SearchResponseData

8.50 and higher

The SearchResponseData class is a data structure delivering the results of a search query.

Namespace

```
Ektron.Cms.Search.SearchResponseData
```

Properties

- `ElapsedTime`. Gets or sets the elapsed execution time for the submitted query.
- `Facets`. Gets or sets a list of Facets corresponding to the search results.
- `PagingInfo`. Gets or sets the paging details for this result set.

- `Results`. Gets or sets a collection of results relevant to the submitted query.
- `SpellingSuggestion`. Gets or sets a spelling suggestion for the submitted query.
- `SuggestedResults`. Gets or sets a collection of suggested results relevant to the submitted query.
- `Terms`. Gets or sets a collection of search terms identified in submitted query.

Constructors

- `SearchResponseData()`. Constructor.

SearchResultData

8.50 and higher

The SearchResultData class represents an individual result item of a search query.

Namespace

```
Ektron.Cms.Search.SearchResultData
```

Properties

- `Item[BooleanMultiValuePropertyExpression]`. Gets and sets a collection of values for the specified boolean property.
- `Item[BooleanPropertyExpression]`. Gets and sets the value for the specified boolean property. If the value of the field is null, false is returned.
- `Item[DateMultiValuePropertyExpression]`. Gets and sets a collection of values for the specified date property.
- `Item[DatePropertyExpression]`. Gets and sets the value for the specified date property. If the value of the field is null, DateTime.MinValue is returned.
- `Item[DecimalMultiValuePropertyExpression]`. Gets and sets a collection of values for the specified decimal property.
- `Item[DecimalPropertyExpression]`. Gets and sets the value for the specified decimal property. If the value of the field is null, 0 is returned.
- `Item[IntegerMultiValuePropertyExpression]`. Gets and sets a collection of values for the specified integer property.
- `Item[IntegerPropertyExpression]`. Gets and sets the value for the specified integer property. If the value of the field is null, 0 is returned.
- `Item[String]`. Gets and sets the value for the specified property. If the value of the field is null, DBNull is returned.
- `Item[StringMultiValuePropertyExpression]`. Gets and sets a collection of values for the specified string property.
- `Item[StringPropertyExpression]`. Gets and sets the value for the specified string property. If the value of the field is null, String.Empty is returned.
- `PropertyCount`. Returns the number of fields in this result item.

- `Similarity`. Gets or sets datac describing how closely this result item relates to other items in the result set.

Constructors

- `SearchResultData()`. Constructor.
- `SearchResultData(DataRow)`. Constructor.
- `SearchResultData(IDictionary<String>,Object>)`. Constructor.

Methods

- `GetValue(BooleanMultiValuePropertyExpression)`. Gets a collection of values associated with the specified property.
- `GetValue(BooleanPropertyExpression)`. Returns the value of the field associated with the specified property.
- `GetValue(DateMultiValuePropertyExpression)`. Gets a collection of values associated with the specified property.
- `GetValue(DatePropertyExpression)`. Returns the value of the field associated with the specified property.
- `GetValue(DecimalMultiValuePropertyExpression)`. Gets a collection of values associated with the specified property.
- `GetValue(DecimalPropertyExpression)`. Returns the value of the field associated with the specified property.
- `GetValue(IntegerMultiValuePropertyExpression)`. Gets a collection of values associated with the specified property.
- `GetValue(IntegerPropertyExpression)`. Returns the value of the field associated with the specified property.
- `GetValue(String)`. Returns the values of the field associated with the specified property. If the value represents a single-value result, the single-value form is returned. If the value represents a multi-value result, the multi-value (collection) form is returned.
- `GetValue(StringMultiValuePropertyExpression)`. Gets a collection of values associated with the specified property.
- `GetValue(StringPropertyExpression)`. Returns the value of the field associated with the specified property.
- `HasColumn(PropertyExpression)`. Returns true if the specified property exists in the result item.
- `HasProperty(String)`. Returns true if the specified property exists in the result item.
- `IsMultiValue(PropertyExpression)`. Returns true if the field associated with the specified property expression represents a collection of values.
- `IsMultiValue(String)`. Returns true if the field associated with the specified property expression represents a collection of values.

- `IsNull(PropertyExpression)`. Returns true if the value of the specified field is null.
- `IsNull(String)`. Returns true if the value of the specified field is null.
- `SetValue(BooleanMultiValuePropertyExpression,IEnumerable<Boolean>)`. Sets a value for the specified boolean property.
- `SetValue(BooleanPropertyExpression,Boolean)`. Sets a value for the specified boolean property.
- `SetValue(DateMultiValuePropertyExpression,IEnumerable<DateTime>)`. Sets a value for the specified date property.
- `SetValue(DatePropertyExpression,DateTime)`. Sets a value for the specified date property.
- `SetValue(DecimalMultiValuePropertyExpression,IEnumerable<Double>)`. Sets a value for the specified decimal property.
- `SetValue(DecimalPropertyExpression,Double)`. Sets a value for the specified decimal property.
- `SetValue(IntegerMultiValuePropertyExpression,IEnumerable<Int64>)`. Sets a collection of values for the specified integer property.
- `SetValue(IntegerPropertyExpression,Int64)`. Sets a value for the specified integer property.
- `SetValue(String,Object)`. Sets a value for the specified property.
- `SetValue(StringMultiValuePropertyExpression,IEnumerable<String>)`. Sets a collection of values for the specified string property.
- `SetValue(StringPropertyExpression,String)`. Sets a value for the specified string property.

Remarks

- `FAST Search for Sharepoint 2010`. Multi-value properties are not supported if your search provider is Microsoft FAST Search for SharePoint 2010. Accessors and methods on this class, accepting a multi-value property expression, should not be used.
- `Search Server 2010`. Multi-value properties are not supported if your search provider is Microsoft Search Server 2010. Accessors and methods on this class, accepting a multi-value property expression, should not be used.
- `Solr`. When accessing data for a multi-value property, your input to the relevant accessors and methods on this class should be a multi-value property expression. A collection of data is returned.

  If multi-value data is accessed with a traditional, single-value, property expression only the first item in that field's collection is returned. Use a multi-value property expression to ensure that you are working with the complete set of data.

If single-value data is accessed with a multi-value property expression an exception is thrown.

SearchSmartFormProperty

8.50 and higher

The SearchSmartFormProperty class is a factory for property expressions targeting Smart Form fields.

Namespace

```
Ektron.Cms.Search.SearchSmartFormProperty
```

Methods

- `GetBooleanProperty(String)`. Gets a `BooleanPropertyExpression` representing the Smart Form field identified by the specified `XPath` value.
- `GetDateProperty(String)`. Gets a `DatePropertyExpression` representing the Smart Form field identified by the specified `XPath` value.
- `GetDecimalProperty(String)`. Gets a `DecimalPropertyExpression` representing the Smart Form field identified by the specified `XPath` value.
- `GetIntegerProperty(String)`. Gets a `IntegerPropertyExpression` representing the Smart Form field identified by the specified `XPath` value.
- `GetStringProperty(String)`. Gets a `StringPropertyExpression` representing the Smart Form field identified by the specified `XPath` value.

Remarks

- `All Providers`. When creating a `PropertyExpression` for an indexed Smart Form field, specify the XPath expression denoting the desired element. (For example: `root/video/description`)

  The XPath expressions for Smart Form fields can be found on the "Display Information" tab when viewing a Smart Form Configuration in Workarea of your site.

- `Solr`. All indexed Smart Form fields are stored as multi-value fields when your search provider is Apache Solr.

  Smart Form fields cannot be sorted on. Apache Solr does not support the sorting of multi-value fields. If sorting behavior is required for data associated with Smart Form content, please consider using a companion metadata definition.

SearchSolrProperty

9.00 and higher

The SearchSolrProperty class exposes enhanced functionality around PropertyExpressions representing Solr index fields. This provides access to behavior unique to the Solr search engine.

Namespace

```
Ektron.Cms.Search.Solr.SearchSolrProperty
```

Properties

- `SearchSolrProperty()` (Static). Constructor.
- `CreateExactStringProperty(StringMultiValuePropertyExpression)`. Creates a property expression referencing the exact-match companion field for the specified property expression. These companion fields provide access to an untokenized and unfiltered representation of CMS property data, which can be particularly useful for performing exact term matches and requesting refinements. Note: Companion fields for Ektron properties are always multi-value fields.
- `CreateExactStringProperty(StringPropertyExpression)`. Creates a property expression referencing the exact-match companion field for the specified property expression. These companion fields provide access to an untokenized and unfiltered representation of CMS property data, which can be particularly useful for performing exact term matches and requesting refinements.

  NOTE: Companion fields for Ektron properties are always multi-value fields.

- `CreateExactStringProperty(String)`. Creates a property expression referencing the exact-match companion field for the specified property expression. These companion fields provide access to an untokenized and unfiltered representation of CMS property data, which can be particularly useful for performing exact term matches and requesting refinements. Note: Companion fields for Ektron properties are always multi-value fields.
- `CreateStringSortProperty(String)`. Creates a property expression referencing the explicit single value companion field for the specified property expression. Note: This is an internal function used to resolve companion index fields (particularly for Smart Form fields) where we must act on a multi-value field as if it were a single value field.
- `CreateIntegerSortProperty(String)`. Creates a property expression referencing the explicit single value companion field for the specified property expression. Note: This is an internal function used to resolve companion index fields (particularly for Smart Form fields) where we must act on a multi-value field as if it were a single value field.
- `CreateDecimalSortProperty(String)`. Creates a property expression referencing the explicit single value companion field for the specified property expression. Note: This is an internal function used to resolve companion index

fields (particularly for Smart Form fields) where we must act on a multi-value field as if it were a single value field.

- `CreateDateSortProperty(String)`. Creates a property expression referencing the explicit single value companion field for the specified property expression. Note: This is an internal function used to resolve companion index fields (particularly for Smart Form fields) where we must act on a multi-value field as if it were a single value field.
- `CreateBooleanSortProperty(String)`. Creates a property expression referencing the explicit single value companion field for the specified property expression. Note: This is an internal function used to resolve companion index fields (particularly for Smart Form fields) where we must act on a multi-value field as if it were a single value field.

SearchType

8.50 and higher

Helper class for generating expressions to limit search result types.

Namespace

```
Ektron.Cms.Search.SearchType
```

Methods

- `IsContent()`. Generates an expression that limits search result type to content.
- `IsDocument()`. Generates an expression that limits search result type to CMS documents. Any valid document types added using the CMS, such as office files, images, PDF or multimedia are included when limiting results by using this property (Please note that valid file types are defined in the Asset Server Setup in the Settings section of Workarea of your site).
- `IsExternal()`. Generates an expression that limits search result type to groups.
- `IsForum()`. Generates an expression that limits search result type to forums.
- `IsGroup()`. Generates an expression that limits search result type to groups.
- `IsMultimedia()`. Generates an expression that limits search result type to multimedia.
- `IsNonUserContent()`. Generates an expression that limits search result type to content.
- `IsProduct()`. Generates an expression that limits search result type to products.
- `IsUser()`. Generates an expression that limits search result type to users.

SearchUserProperty

8.50 and higher

The UserProperty class provides access to the default PropertyExpressions for formulating user queries.

Namespace

```
Ektron.Cms.Search.SearchUserProperty
```

Properties

- `Avatar`. Gets the PropertyExpression for the user avatar field.
- `DateModified`. Gets the PropertyExpression for the date modified field.
- `DisplayName`. Gets the PropertyExpression for the display name field.
- `EmailAddress`. Gets the PropertyExpression for the email address field.
- `FirstName`. Gets the PropertyExpression for the first name field.
- `HighlightedSummary`. Gets the PropertyExpression for the highlighted summary field.
- `Id`. Gets the PropertyExpression for the user ID field.
- `Language`. Gets the PropertyExpression for the user language field.
- `LastName`. Gets the PropertyExpression for the last name field.
- `MapAddress`. Gets the PropertyExpression for the map address field.
- `MapLatitude`. Gets the PropertyExpression for the map latitude field.
- `MapLongitude`. Gets the PropertyExpression for the map longitude field.
- `MembershipUser`. Gets the PropertyExpression for the membership user field.
- `Path`. Gets the PropertyExpression for the path field.
- `PrivateProfile`. Gets the PropertyExpression for the private profile field.
- `QuickLink`. Gets the PropertyExpression for the QuickLink field.
- `Rank`. Gets the PropertyExpression for the rank field.
- `Size`. Gets the PropertyExpression for the content size field.
- `Tags`. Gets the PropertyExpression for the user tags field.
- `TaxonomyCategory`. Gets the PropertyExpression for the taxonomy category field.
- `Type`. Gets the PropertyExpression for the user type field.
- `UserFriends`. Gets the PropertyExpression for the user fiends field.
- `UserName`. Gets the PropertyExpression for the user name field.

Remarks

- `All Providers`. The `HighlightedSummary` property expression is used to request a dynamically generated summary for each result item in your result set. This behavior varies from search provider to search provider. Its effectiveness also varies based on the quantity and quality of the data available in the index for a given piece of content. For example, XML-based content, such as Smart Form data, is less likely to render a meaningful summary than

traditional textual content. It is recommended that static summaries are conditionally displayed for result items in cases where the dynamically generated one is insufficient.

- `Solr`. The `TaxonomyCategory` property is a multi-value field in Solr. Multi-value fields provide a better development experience when consuming field data representing a collection of values. It is recommended that you reference the `TaxonomyCategory` property as a multi-value property expression to ensure that you are always working with the complete data for your result item (See: StringMultiValuePropertyExpression class):

  ```
  SearchContentProperty.TaxonomyCategory.AsMultiValue()
  ```

  General keyword searches of hierarchical field data is not supported. Fields such as `TaxonomyCategory`, `FolderPath`, and `FolderIdPath` have been optimized for queries that are programmatically constructed. When querying this type of data, your input should be a rooted, fully formed, path. For example, consider a taxonomy category of "\Departments\HR\Benefits". A query, against the taxonomy category field, with an input of "\Departments\HR" would return results associated with the category. A query with an input of "HR" would not.

  Solr will analyze the first 100,000 characters of an individual document when generating a `HighlightedSummary` value. For large content, a dynamic summary will not be generated if the search terms for your query are not found in the first 100,000 characters of the document.

StringFacet

8.60 and higher

The StringFacet class represents a classification of search results on a string index field.

Namespace

```
Ektron.Cms.Search.StringFacet
```

Properties

- `Buckets`. Gets or sets the collection of facet buckets associated with this facet.
- `Property`. Gets or sets the property associated with this facet.

Constructors

- `StringFacet(PropertyExpression)`. Constructor.
- `StringFacet (PropertyExpression,ICollection<UniqueFacetBucket<String>>)`. Constructor.
- `StringFacet(StringMultiValuePropertyExpression)`. Constructor.

- `StringFacet (StringMultiValuePropertyExpression,ICollection<UniqueFacetBucket <String>>)`. Constructor.
- `StringFacet(StringPropertyExpression)`. Constructor.
- `StringFacet (StringPropertyExpression,ICollection<UniqueFacetBucket<String>>)` . Constructor.

StringRefinementSpecification

8.60 and higher

The StringRefinementSpecification represents a strongly typed refinement specification for string index properties.

Namespace

```
Ektron.Cms.Search.StringRefinementSpecification
```

Properties

- `Filter`. Hidden until this functionality becomes available.
- `Order`. Hidden until this functionality becomes available.
- `Property`. Gets or sets the refinement property.

Constructors

- `StringRefinementSpecification()`. Constructor.
- `StringRefinementSpecification (StringMultiValuePropertyExpression)`. Constructor.
- `StringRefinementSpecification (StringMultiValuePropertyExpression,RefinementOrderData)`. Constructor.
- `StringRefinementSpecification (StringMultiValuePropertyExpression,String)`. Constructor.
- `StringRefinementSpecification (StringMultiValuePropertyExpression,String,RefinementOrderData)`. Constructor.
- `StringRefinementSpecification(StringPropertyExpression)`. Constructor.
- `StringRefinementSpecification (StringPropertyExpression,RefinementOrderData)`. Constructor.
- `StringRefinementSpecification(StringPropertyExpression,String)`. Constructor.
- `StringRefinementSpecification (StringPropertyExpression,String,RefinementOrderData)`. Constructor.

Remarks

- `All Providers`. Add a `StringRefinementSpecification` to search criteria to identify a field for which facet data should be returned with the query results.
- `Search Server 2010`. The `StringRefinementSpecification` is not applicable if your search provider is Microsoft Search Server 2010. Deep refinement of result sets is not available.
- `Solr`. The terms used to generate string facets reflect the tokenization and filtering applied to the target index field when your search provider is Apache Solr. Often, it is better to reference a string field with minimal tokenization and filtering to achieve the best experience.

  All indexed Ektron properties have an exact-match companion field, which provides access to an untokenized and unfiltered representation of the property's data. When requesting string-based refinements, these fields often provide the desired experience. (To access these fields, see the `SearchSolrProperty` class.)

SimilarityResponseData

8.60 and higher

The SimilarityResponseData class encapsulates information describing how closely a result item relates to other items in the result set.

Namespace

```
Ektron.Cms.Search.SimilarityResponseData
```

Properties

- `SimilarityVector`. Gets or sets a collection of similar tokens that provide document term relevancy data.

Constructors

- `SimilarityResponseData()`. Constructor.

SimilaritySearchCriteria

8.60 and higher

The SimilaritySearchCriteria class captures metadata associated with a particular result item. This data is used to refine the parent query to a set of results that are similar to that result item.

Namespace

```
Ektron.Cms.Search.SimilaritySearchCriteria
```

Properties

- `IsEnabled`. Gets or sets a flag indicating whether or not similarity search should be enabled for the parent query.
- `QueryType`. Gets or sets the type of similarity query to be issued.
- `SimilarityVector`. Gets a document vector describing a particular indexed item which is to be evaluated by the search provider, identifying other similar documents.

Constructors

- `SimilaritySearchCriteria()`. Constructor.

SuggestedResultsData

8.50 and higher

Defines a suggested search result.

Namespace

```
Ektron.Cms.Search.SuggestedResultData
```

Properties

- `Description`. Gets and sets the description of the suggested result.
- `Title`. Gets and sets the title of the suggested result.
- `Url`. Gets and sets the URL of the suggested result.

Methods

- `Equals(Object)`. Returns true if the specified suggested result is equal to this instance.
- `GetHashCode()`. Returns the hash code for this instance of `SuggestedResultData`.
- `ToString()`. Returns the string representation of this instance of `SuggestedResultData`.

UniqueFacetBucket

8.60 and higher

The UniqueFacetBucket class describes a facet bucket representing a single unique value.

Namespace

Ektron.Cms.Search.UniqueFacetBucket<T>

Properties

- `Value`. Gets or sets the value associated with this facet bucket.

Constructors

- `UniqueFacetBucket(T,Int64)`. Constructor.

Search Exceptions

AllNoiseException

8.60 and higher

The AllNoiseException occurs when a query submitted for search consists entirely of noise words.

Namespace

```
Ektron.Cms.Search.AllNoiseException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.

Methods

- `AllNoiseException()`. Constructor.
- `AllNoiseException(Exception)`. Constructor.
- `AllNoiseException(String)`. Constructor.
- `AllNoiseException(String,Exception)`. Constructor.

DuplicateSuggestedResultException

8.50 and higher

A `DuplicateSuggestedResultException` occurs when an attempt is made to save a suggested result that already exists in the system.

Namespace

```
Ektron.Cms.Search.DuplicateSuggestedResultException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.
- `Duplicates`. Gets or sets a dictionary of duplicate terms (mapped to the primary phrase with which they are associated).

Methods

- `DuplicateSuggestedResultException()`. Constructor.
- `DuplicateSuggestedResultException(Exception)`. Constructor.
- `DuplicateSuggestedResultException(String)`. Constructor.
- `DuplicateSuggestedResultException(String,Exception)`. Constructor.

DuplicateSynonymException

8.50 and higher

A DuplicateSynonymException occurs when an attempt is made to save a synonym that already exists in the system.

Namespace

```
Ektron.Cms.Search.DuplicateSynonymException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.

Methods

- `DuplicateSynonymException()`. Constructor.
- `DuplicateSynonymException(Exception)`. Constructor.
- `DuplicateSynonymException(String)`. Constructor.
- `DuplicateSynonymException(String,Exception)`. Constructor.

EmptyQueryException

8.50 and higher

The `EmptyQueryException` is thrown when criteria submitted for search contains an empty query.

Namespace

```
Ektron.Cms.Search.EmptyQueryException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.

Methods

- `EmptyQueryException()`. Constructor.
- `EmptyQueryException(Exception)`. Constructor.
- `EmptyQueryException(String)`. Constructor.
- `EmptyQueryException(String,Exception)`. Constructor.

EmptyReturnPropertiesException

8.50 and higher

The `EmptyReturnPropertiesException` occurs when a query criteria is submitted with an empty collection of return properties.

Namespace

```
Ektron.Cms.Search.EmptyReturnPropertiesException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.

Methods

- `EmptyReturnPropertiesException()`. Constructor.
- `EmptyReturnPropertiesException(Exception)`. Constructor.
- `EmptyReturnPropertiesException(String)`. Constructor.
- `EmptyReturnPropertiesException(String,Exception)`. Constructor.

InvalidOrderByException

8.50 and higher

The `InvalidOrderByException` occurs when the OrderBy criteria is applied in a manner that is not supported by the search provider.

Namespace

```
Ektron.Cms.Search.InvalidOrderByException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.

Methods

- `InvalidOrderByException()`. Constructor.
- `InvalidOrderByException(Exception)`. Constructor.
- `InvalidOrderByException(String)`. Constructor.
- `InvalidOrderByException(String,Exception)`. Constructor.

InvalidPropertyException

8.50 and higher

The `InvalidPropertyException` occurs when a property referenced in the query does not exist in the system or is used in a manner which is not supported.

Namespace

```
Ektron.Cms.Search.InvalidPropertyException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.

Methods

- `InvalidPropertyException()`. Constructor.
- `InvalidPropertyException(Exception)`. Constructor.
- `InvalidPropertyException(String)`. Constructor.
- `InvalidPropertyException(String,Exception)`. Constructor.

InvalidScopeException

8.50 and higher

The `InvalidScopeException` occurs when the scope specified in the query criteria is invalid or does not exist.

Namespace

```
Ektron.Cms.Search.InvalidScopeException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.

Methods

- `InvalidScopeException()`. Constructor.
- `InvalidScopeException(Exception)`. Constructor.

- `InvalidScopeException(String)`. Constructor.
- `InvalidScopeException(String,Exception)`. Constructor.

MalformedExpressionException

8.50 and higher

The `MalformedExpressionException` is raised when an invalid expression is used in a query.

Namespace

```
Ektron.Cms.Search.MalformedExpressionException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.

Methods

- `MalformedExpressionException()`. Constructor.
- `MalformedExpressionException(Exception)`. Constructor.
- `MalformedExpressionException(Exception,String)`. Constructor.
- `MalformedExpressionException(String)`. Constructor.
- `MalformedExpressionException(String,Exception)`. Constructor.

NoResultsException

8.50 and higher

The `NoResultsException` occurs when a query yields no results.

Namespace

```
Ektron.Cms.Search.NoResultsException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.

Methods

- `NoResultsException()`. Constructor.
- `NoResultsException(Exception)`. Constructor.
- `NoResultsException(String)`. Constructor.
- `NoResultsException(String,Exception)`. Constructor.

SearchAuthorizationException

8.50 and higher

The `SearchAuthorizationException` occurs when authentication with the search system fails.

Namespace

```
Ektron.Cms.Search.SearchAuthorizationException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.

Methods

- `SearchAuthorizationException()`. Constructor.
- `SearchAuthorizationException(Exception)`. Constructor.
- `SearchAuthorizationException(String)`. Constructor.
- `SearchAuthorizationException(String,Exception)`. Constructor.

SearchException

8.50 and higher

The `SearchException` indicates that an error has occurred while performing a search-related activity.

Namespace

```
Ektron.Cms.Search.SearchException
```

Properties

- `Details`. Gets or sets any additional details associate with this exception.

Methods

- `SearchException()`. Constructor.
- `SearchException(Exception)`. Constructor.
- `SearchException(String)`. Constructor.
- `SearchException(String,Exception)`. Constructor.

Search Expressions

AndExpression

8.60 and higher

Defines an expression representing an "and" operator.

Namespace

```
Ektron.Cms.Search.Expressions.AndExpression
```

Properties

- `Left`. Gets the left branch of the binary expression tree.
- `Right`. Gets the right branch of the binary expression tree.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to process the current expression.
- `AndExpression()`. Constructor.
- `AndExpression(Expression,Expression)`. Constructor.

BinaryExpression

8.50 and higher

Defines an expression representing a binary operator.

Namespace

```
Ektron.Cms.Search.Expressions.BinaryExpression
```

Properties

- `Left`. Gets the left branch of the binary expression tree.
- `Right`. Gets the right branch of the binary expression tree.

Methods

- `ValidateComparisonExpression(Expression,Expression)`. Validates a comparison expression in which the 'left' expression must be of type PropertyExpression and the 'right' expression must be of type ValueExpression.

BooleanMultiValuePropertyExpression

9.00 and higher

Defines an expression representing a boolean index field which may contain multiple values.

Namespace

```
Ektron.Cms.Search.Expressions.BooleanMultiValuePropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `Name`. Gets or sets the internal property name.
- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `BooleanMultiValuePropertyExpression(String)`. Constructor.
- `Contains(Boolean)`. Constructs a ContainsExpression using the current object as its left branch and the specified value as its right branch.
- `IsNotNull()`. Creates an 'IsNull' expression. (Not supported)
- `IsNull()`. Creates an 'IsNotNull' expression. (Not supported)

Remarks

- `FAST Search for Sharepoint 2010`. Multi-value properties are not supported when your search provider is Microsoft FAST Search for SharePoint 2010.
- `Search Server 2010`. Multi-value properties are not supported when your search provider is Microsoft Search Server 2010.
- `Solr`. Field names in Apache Solr are case-sensitive. When constructing a `PropertyExpression` for a custom Solr index field, ensure that the proper casing is applied.

BooleanPropertyExpression

8.50 and higher

Defines an expression representing a boolean index field.

Namespace

```
Ektron.Cms.Search.Expressions.BooleanPropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `Name`. Gets or sets the internal property name.
- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to process the current expression.
- `AsMultiValue()`. Returns a reference to the multi-value form of this property expression, typically used to retrieve a collection of values from the index.

  NOTE: An index field must be specifically configured to process multi-value data for this to be applicable. Not all search providers support this behavior.

- `BooleanPropertyExpression(String)`. Constructor.
- `EqualTo(Boolean)`. Constructs an EqualsExpression using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `NotEqualTo(Boolean)`. Constructs an NotEqualsExpression using the current object as its left-branch, and the given parameter "value" as its right-branch.

Remarks

- `Solr`. Field names in Apache Solr are case sensitive. When constructing a `PropertyExpression` for a custom Solr index field, ensure that the proper casing is applied.

BooleanValueExpression

8.50 and higher

Defines an expression representing a boolean value.

Namespace

```
Ektron.Cms.Search.Expressions.BooleanValueExpression
```

Properties

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.

ContainsExpression

8.50 and higher

Defines an expression representing a "contains" operator.

Namespace

```
Ektron.Cms.Search.Expressions.ContainsExpression
```

Properties

- `Forms`. Gets or sets which alternative word forms should also be considered when searching for the specified phrase.
- `Phrase`. Gets or sets the search phrase for this expression.
- `Property`. Gets the property to evaluate.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `ContainsExpression()`. Constructor.
- `ContainsExpression(BooleanMultiValuePropertyExpression,Boolean)`. Constructor.
- `ContainsExpression(DateMultiValuePropertyExpression,DateTime)`. Constructor.
- `ContainsExpression(DecimalMultiValuePropertyExpression,Double)`. Constructor.
- `ContainsExpression(IntegerMultiValuePropertyExpression,Int64)`. Constructor.
- `ContainsExpression(PropertyExpression,Expression)`. Constructor.
- `ContainsExpression(PropertyExpression,Expression,WordForms)`. Constructor.
- `ContainsExpression(String)`. Constructor.
- `ContainsExpression(StringMultiValuePropertyExpression,String)`. Constructor.
- `ContainsExpression(StringPropertyExpression,String)`. Constructor.
- `ContainsExpression (StringPropertyExpression,StringValueExpression)`. Constructor.
- `ContainsExpression(StringPropertyExpression,String,WordForms)`. Constructor.
- `ContainsExpression(String,WordForms)`. Constructor.

Remarks

- `All Providers`. Use caution when including special characters in your expressions as your search provider may interpret these as wildcards. This can have an unintended performance impact. Consult your search provider vendor's documentation for information regarding how to safely escape these characters.
- `FAST Search for Sharepoint 2010`. The `WordForms` parameter is not supported when your search provider is Microsoft FAST Search for SharePoint. Instead, use the fields exposed directly on the KeywordSearchCriteria class (EnableStemming, and so on) to modify this behavior.

- `Solr`. The `WordForms` parameter is not supported when your search provider is Apache Solr. Stemming and other similar features cannot be toggled at query-time. They are configured on a field by field basis in the `schema.xml` for a given Solr core. If a different behavior is required, you should consider customizing the index schema for your site.

DateMultiValuePropertyExpression

9.00 and higher

Defines an expression representing a date index field which may contain multiple values.

Namespace

```
Ektron.Cms.Search.Expressions.DateMultiValuePropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `Name`. Gets or sets the internal property name.
- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `Contains(DateTime)`. Constructs a ContainsExpression using the current object as its left branch and the specified value as its right branch.
- `DateMultiValuePropertyExpression(String)`. Constructor.
- `IsNotNull()`. Creates an 'IsNull' expression. (Not supported)
- `IsNull()`. Creates an 'IsNotNull' expression. (Not supported)

Remarks

- `FAST Search for Sharepoint 2010`. Multi-value properties are not supported when your search provider is Microsoft FAST Search for SharePoint 2010.
- `Search Server 2010`. Multi-value properties are not supported when your search provider is Microsoft Search Server 2010.
- `Solr`. Field names in Apache Solr are case sensitive. When constructing a `PropertyExpression` for a custom Solr index field, ensure that the proper casing is applied.

DatePropertyExpression

8.50 and higher

Defines an expression representing a date index field.

Namespace

```
Ektron.Cms.Search.Expressions.DatePropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `Name`. Gets or sets the internal property name.
- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node
- `AsMultiValue()`. Returns a reference to the multi-value form of this property expression, typically used to retrieve a collection of values from the index.

  NOTE: An index field must be specifically configured to process multi-value data for this to be applicable. Not all search providers support this behavior.

- `DatePropertyExpression(String)`. Constructor.
- `EqualTo(DateTime)`. Constructs an `EqualToExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `GreaterThan(DateTime)`. Constructs a `GreaterThanExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `GreaterThanOrEqualTo(DateTime)`. Constructs a `GreaterThanOrEqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `LessThan(DateTime)`. Constructs an `LessThanExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `LessThanOrEqualTo(DateTime)`. Constructs an `LessThanOrEqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `NotEqualTo(DateTime)`. Constructs an `NotEqualToExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.

Remarks

- `Solr`. Field names in Apache Solr are case-sensitive. When constructing a `PropertyExpression` for a custom Solr index field, ensure that the proper casing is applied.

DateValueExpression

8.50 and higher

Defines an expression representing a date value.

Namespace

```
Ektron.Cms.Search.Expressions.DateValueExpression
```

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.

DecimalMultiValuePropertyExpression

9.00 and higher

Defines an expression representing a decimal index field which may contain multiple values.

Namespace

```
Ektron.Cms.Search.Expressions.DecimalMultiValuePropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `Name`. Gets or sets the internal property name.
- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `Contains(Double)`. Constructs a ContainsExpression using the current object as its left branch and the specified value as its right branch.
- `DecimalMultiValuePropertyExpression(String)`. Constructor.
- `IsNotNull()`. Creates an 'IsNull' expression. (Not supported)
- `IsNull()`. Creates an 'IsNotNull' expression. (Not supported)

Remarks

- `FAST Search for Sharepoint 2010`. Multi-value properties are not supported when your search provider is Microsoft FAST Search for SharePoint 2010.
- `Search Server 2010`. Multi-value properties are not supported when your search provider is Microsoft Search Server 2010.

- `Solr`. Field names in Apache Solr are case-sensitive. When constructing a PropertyExpression for a custom Solr index field, ensure that the proper casing is applied.

DecimalPropertyExpression

8.50 and higher

Defines an expression representing a decimal index field.

Namespace

```
Ektron.Cms.Search.Expressions.DecimalPropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `Name`. Gets or sets the internal property name.
- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `AsMultiValue()`. Returns a reference to the multi-value form of this property expression, typically used to retrieve a collection of values from the index.

  NOTE: An index field must be specifically configured to process multi-value data for this to be applicable. (Not all search providers support this behavior.)

- `DecimalPropertyExpression(String)`. Constructor.
- `EqualTo(Double)`. Constructs an `EqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `GreaterThan(Double)`. Constructs a `GreaterThanExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `GreaterThanOrEqualTo(Double)`. Constructs a `GreaterThanOrEqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `LessThan(Double)`. Constructs a `LessThanExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `LessThanOrEqualTo(Double)`. Constructs a `LessThanOrEqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `NotEqualTo(Double)`. Constructs a `NotEqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.

Remarks

- `Solr`. Field names in Apache Solr are case sensitive. When constructing a `PropertyExpression` for a custom Solr index field, ensure that the proper casing is applied.

DecimalValueExpression

8.50 and higher

Defines an expression representing a decimal value.

Namespace

```
Ektron.Cms.Search.Expressions.DecimalValueExpression
```

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.

EqualsExpression

8.50 and higher

Defines an expression representing an "equals" operator.

Namespace

```
Ektron.Cms.Search.Expressions.EqualsExpression
```

Properties

- `Left`. Gets the left branch of the binary expression tree.
- `Right`. Gets the right branch of the binary expression tree.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `EqualsExpression()`. Constructor.
- `EqualsExpression(BooleanPropertyExpression,Boolean)`. Constructor.
- `EqualsExpression(DatePropertyExpression,DateTime)`. Constructor.
- `EqualsExpression(DecimalPropertyExpression,Double)`. Constructor.
- `EqualsExpression(Expression,Expression)`. Constructor.
- `EqualsExpression(IntegerPropertyExpression,Int64)`. Constructor.
- `EqualsExpression(StringPropertyExpression,String)`. Constructor.

Remarks

- `All Providers`. Use the `EqualsExpression` with caution when applied to string properties. Any variance between the field value and the target value will result in a non-match. Often, a `ContainsExpression` is more appropriate.

  The precision of decimal fields can vary from provider to provider. Consider querying a range of values rather than applying an exact equality statement in these cases.

- `Solr`. The `EqualsExpression` is evaluated in a case sensitive manner.

  Exact-match comparisons for text fields (`StringPropertyExpression` and `StringMultiValuePropertyExpression`) are given special handling. String equality of tokenized textual data is not supported in Solr. Instead, exact-match comparisons must be performed against an untokenized version of the text data. To achieve the appropriate behavior, the Ektron search API will automatically re-target string equality expressions to an untokenized companion field.

  When adding a custom text field to your site's schema, you should also add an exact match companion field. The companion field is required for any literal string comparisons (string equality) appearing in your queries. Failure to add this companion field will result in an error if the `EqualsExpression` appears in a query. (See the `schema.xml` for your site for additional information.)

Expression

8.50 and higher

Abstract class defining the base logical search expression.

Namespace

```
Ektron.Cms.Search.Expressions.Expression
```

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `And(Expression)`. Returns an AND expression of this expression and the specified right expression.
- `And(IEnumerable<Expression>)`. Returns a list of terms ANDed together.
- `op_BitwiseAnd(Expression,Expression)`. Returns an AND expression for the specified left and right expressions.
- `op_BitwiseOr(Expression,Expression)`. Returns an OR expression for the specified left and right expressions.
- `op_LogicalNot(Expression)`. Returns an expression negating the specified expression.

- `Or(Expression)`. Returns an OR expression of this expression and the specified right expression.
- `Or(IEnumerable<Expression>)`. Returns a list of terms ORed together.

GreaterThanExpression

8.50 and higher

Defines an expression representing a "greater-than" operator.

Namespace

```
Ektron.Cms.Search.Expressions.GreaterThanExpression
```

Properties

- `Left`. Gets the left branch of the binary expression tree.
- `Right`. Gets the right branch of the binary expression tree.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `GreaterThanExpression()`. Constructor.
- `GreaterThanExpression(DatePropertyExpression,DateTime)`. Constructor.
- `GreaterThanExpression(DecimalPropertyExpression,Double)`. Constructor.
- `GreaterThanExpression(Expression,Expression)`. Constructor.
- `GreaterThanExpression(IntegerPropertyExpression,Int64)`. Constructor.
- `GreaterThanExpression(StringPropertyExpression,String)`. Constructor.

GreaterThanOrEqualsExpression

8.50 and higher

Defines an expression representing a "greater-than-or-equals" operator.

Namespace

```
Ektron.Cms.Search.Expressions.GreaterThanOrEqualsExpression
```

Properties

- `Left`. Gets the left branch of the binary expression tree.
- `Right`. Gets the right branch of the binary expression tree.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `GreaterThanOrEqualsExpression()`. Constructor.
- `GreaterThanOrEqualsExpression(DatePropertyExpression,DateTime)`. Constructor.
- `GreaterThanOrEqualsExpression(DecimalPropertyExpression,Double)`. Constructor.
- `GreaterThanOrEqualsExpression(Expression,Expression)`. Constructor.
- `GreaterThanOrEqualsExpression(IntegerPropertyExpression,Int64)`. Constructor.
- `GreaterThanOrEqualsExpression(StringPropertyExpression,String)`. Constructor.

IntegerMultiValuePropertyExpression

9.00 and higher

Defines an expression representing a integer index field which may contain multiple values.

Namespace

```
Ektron.Cms.Search.Expressions.IntegerMultiValuePropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `Name`. Gets or sets the internal property name.
- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `Contains(Int64)`. Constructs a ContainsExpression using the current object as its left branch and the specified value as its right branch.
- `IntegerMultiValuePropertyExpression(String)`. Constructor.
- `IsNotNull()`. Creates an 'IsNull' expression. (Not supported)
- `IsNull()`. Creates an 'IsNotNull' expression. (Not supported)

Remarks

- `FAST Search for Sharepoint 2010`. Multi-value properties are not supported when your search provider is Microsoft FAST Search for SharePoint 2010.
- `Search Server 2010`. Multi-value properties are not supported when your search provider is Microsoft Search Server 2010.
- `Solr`. Field names in Apache Solr are case sensitive. When constructing a `PropertyExpression` for a custom Solr index field, ensure that the proper casing is applied.

IntegerPropertyExpression

8.50 and higher

Defines an expression representing a integer index field.

Namespace

```
Ektron.Cms.Search.Expressions.IntegerPropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `Name`. Gets or sets the internal property name.
- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node
- `AsMultiValue()`. Returns a reference to the multi-value form of this property expression, typically used to retrieve a collection of values from the index.

  NOTE: An index field must be specifically configured to process multi-value data for this to be applicable. Not all search providers support this behavior.

- `EqualTo(Int64)`. Constructs an `EqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `GreaterThan(Int64)`. Constructs a `GreaterThanExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `GreaterThanOrEqualTo(Int64)`. Constructs a `GreaterThanOrEqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `IntegerPropertyExpression(String)`. Constructor.
- `LessThan(Int64)`. Constructs a `LessThanExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.

- `LessThanOrEqualTo(Int64)`. Constructs a `LessThanOrEqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `NotEqualTo(Int64)`. Constructs a `NotEqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.

Remarks

- `Solr`. Field names in Apache Solr are case sensitive. When constructing a `PropertyExpression` for a custom Solr index field, ensure that the proper casing is applied.

IntegerValueExpression

8.50 and higher

Defines an expression representing an integer value.

Namespace

```
Ektron.Cms.Search.Expressions.IntegerValueExpression
```

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node
- `IntegerValueExpression(Int64)`. Constructor.

KeywordExpression

8.50 and higher

Defines an expression representing a "keyword-search" operator.

Namespace

```
Ektron.Cms.Search.Expressions.KeywordExpression
```

Properties

- `Mode`. Gets or sets the mode for this keyword expression, which indicates whether individual terms are implicitly ANDed or ORed.
- `Phrase`. Gets or sets the StringValueExpression representing the keyword text of this expression.
- `Property`. Gets the property to which this KeywordExpression applies.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `KeywordExpression()`. Constructor.
- `KeywordExpression(PropertyExpression,String,KeywordMode)`. Constructor.
- `KeywordExpression(String,KeywordMode)`. Constructor.

Remarks

- `Solr`. The KeywordMode option is not supported if your search provider is Apache Solr. KeywordExpressions with multiple terms assume an implicit AND between terms.

LessThanExpression

8.50 and higher

Defines an expression representing a "less-than" operator.

Namespace

```
Ektron.Cms.Search.Expressions.LessThanExpression
```

Properties

- `Left`. Gets the left branch of the binary expression tree.
- `Right`. Gets the right branch of the binary expression tree.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `LessThanExpression()`. Constructor.
- `LessThanExpression(DatePropertyExpression,DateTime)`. Constructor.
- `LessThanExpression(DecimalPropertyExpression,Double)`. Constructor.
- `LessThanExpression(Expression,Expression)`. Constructor.
- `LessThanExpression(IntegerPropertyExpression,Int64)`. Constructor.
- `LessThanExpression(StringPropertyExpression,String)`. Constructor.

LessThanOrEqualsExpression

8.50 and higher

Defines an expression representing a "less-than-or-equals" operator.

Namespace

```
Ektron.Cms.Search.Expressions.LessThanOrEqualsExpression
```

Properties

- `Left`. Gets the left branch of the binary expression tree.
- `Right`. Gets the right branch of the binary expression tree.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `LessThanOrEqualsExpression()`. Constructor.
- `LessThanOrEqualsExpression(DatePropertyExpression,DateTime)`. Constructor.
- `LessThanOrEqualsExpression(DecimalPropertyExpression,Double)`. Constructor.
- `LessThanOrEqualsExpression(Expression,Expression)`. Constructor.
- `LessThanOrEqualsExpression(IntegerPropertyExpression,Int64)`. Constructor.
- `LessThanOrEqualsExpression(StringPropertyExpression,String)`. Constructor.

NotEqualsExpression

8.50 and higher

Defines an expression representing a "not-equals" operator.

Namespace

```
Ektron.Cms.Search.Expressions.NotEqualsExpression
```

Properties

- `Left`. Gets the left branch of the binary expression tree.
- `Right`. Gets the right branch of the binary expression tree.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `NotEqualsExpression()`. Constructor.
- `NotEqualsExpression(BooleanPropertyExpression,Boolean)`. Constructor.
- `NotEqualsExpression(DatePropertyExpression,DateTime)`. Constructor.
- `NotEqualsExpression(DecimalPropertyExpression,Double)`. Constructor.

- `NotEqualsExpression(Expression,Expression)`. Constructor.
- `NotEqualsExpression(IntegerPropertyExpression,Int64)`. Constructor.
- `NotEqualsExpression(StringPropertyExpression,String)`. Constructor.

NotExpression

8.50 and higher

Defines an expression representing a "not" operator.

Namespace

```
Ektron.Cms.Search.Expressions.NotExpression
```

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `NotExpression(Expression)`. Constructor.

OrExpression

8.50 and higher

Defines an expression representing an "or" operator.

Namespace

```
Ektron.Cms.Search.Expressions.OrExpression
```

Properties

- `Left`. Gets the left branch of the binary expression tree.
- `Right`. Gets the right branch of the binary expression tree.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `OrExpression(Expression,Expression)`. Constructor.

PropertyExpression

8.50 and higher

The `PropertyExpression` class serves as a base for expressions identifying a specific searchable property.

Namespace

```
Ektron.Cms.Search.Expressions.PropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `Name`. Gets or sets the internal property name.
- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `And(Expression)`. Returns an AND expression of this expression and the specified right expression.

  NOTE: The AND expression is not support for expressions of type `PropertyExpression`.

- `CreateBooleanMultiValueProperty(String,PropertyCategory)`. Creates a `BooleanMultiValuePropertyExpression` for the specified searchable property.
- `CreateBooleanProperty(String)`. Creates a `BooleanPropertyExpression` for the specified searchable property.
- `CreateBooleanProperty(String,PropertyCategory)`. Creates a `BooleanPropertyExpression` for the specified searchable property.
- `CreateDateMultiValueProperty(String,PropertyCategory)`. Creates a `DateMultiValuePropertyExpression` for the specified searchable property.
- `CreateDateProperty(String)`. Creates a DatePropertyExpression for the specified searchable property.
- `CreateDateProperty(String,PropertyCategory)`. Creates a `DatePropertyExpression` for the specified searchable property.
- `CreateDecimalMultiValueProperty(String,PropertyCategory)`. Creates a `DecimalMultiValuePropertyExpression` for the specified searchable property.
- `CreateDecimalProperty(String)`. Creates a `DecimalPropertyExpression` for the specified searchable property.
- `CreateDecimalProperty(String,PropertyCategory)`. Creates a `DecimalPropertyExpression` for the specified searchable property.
- `CreateIntegerProperty(String)`. Creates a `IntegerPropertyExpression` for the specified searchable property.
- `CreateIntegerMultiValueProperty(String,PropertyCategory)`. Creates a `IntegerMultiValuePropertyExpression` for the specified searchable property.
- `CreateIntegerProperty(String,PropertyCategory)`. Creates a `IntegerPropertyExpression` for the specified searchable property.

- `CreateRankProperty()`. Creates a `RankPropertyExpression` for the default rank field.
- `CreateRankProperty(String)`. Creates a `RankProperty` expression for the default rank field with the specified rank ID.
- `CreateRankProperty(String,String)`. Creates a `RankProperty` expression for the specified rank field and rank ID.
- `CreateStringMultiValueProperty(String,PropertyCategory)`. Creates a `StringMultiValuePropertyExpression` for the specified searchable property.
- `CreateStringProperty(String)`. Creates a `StringPropertyExpression` for the specified searchable property.
- `CreateStringProperty(String,PropertyCategory)`. Creates a `StringPropertyExpression` for the specified searchable property.
- `Equals(Object)`. Returns true if this `PropertyExpression` instance is equal to the specified one.
- `GetHashCode()`. Gets the hash code for this `PropertyExpression` instance.
- `IsNull()`. Creates an 'IsNull' expression.
- `IsNotNull()`. Creates an 'IsNotNull' expression.
- `Or(Expression)`. Returns an OR expression of this expression and the specified right expression.

  NOTE: The OR expression is not support for expressions of type `PropertyExpression`.

- `PropertyExpression(String)`. Constructor.
- `ToString()`. Gets the string representation of this PropertyExpression instance.

Remarks

- `Solr`. Field names in Apache Solr are case sensitive. When constructing a `PropertyExpression` for a custom Solr index field, ensure that the proper casing is applied.

RankPropertyExpression

8.60 and higher

Defines an expression representing a rank field in the index.

Namespace

```
Ektron.Cms.Search.Expressions.RankPropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `IsDefault`. Gets or sets a flag indicating whether the default ranking profile is applied.

- `Name`. Gets or sets the internal property name.
- `RankID`. Gets or sets the ID of the ranking profile to be applied.
- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `RankPropertyExpression()`. Constructor.
- `RankPropertyExpression(String)`. Constructor.
- `RankPropertyExpression(String,String)`. Constructor.

Remarks

- `FAST Search for Sharepoint 2010`. The `RankID` identifies the rank profile to be applied when calculating the relevance of documents matching your query.
- `Search Server 2010`. The `RankID` property is not supported for Microsoft Search Server.
- `Solr`. The `RankID` property is not supported for Apache Solr.

ScopeExpression

8.50 and higher

A `ScopeExpression` identifies a subset of a search index to be queried.

Namespace

```
Ektron.Cms.Search.Expressions.ScopeExpression
```

Properties

- `Default`. Gets a `ScopeExpression` identifying the default scope for the current site.

Methods

- `Accept(ExpressionVisitor)`. Accepts an expression visitor capable of translating this expression into a search query.
- `ScopeExpression()`. Implicit conversion of a single `ScopeExpression` to a list of ScopeExpressions.
- `ScopeExpression(String)`. Constructor.

StringMultiValuePropertyExpression

9.00 and higher

Defines an expression representing a string index field which may contain multiple values.

Namespace

```
Ektron.Cms.Search.Expressions.StringMultiValuePropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `Name`. Gets or sets the internal property name.
- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `Contains(String)`. Constructs a ContainsExpression using the current object as its left branch and the specified phrase as its right branch.
- `IsNotNull()`. Creates an 'IsNull' expression. (Not supported)
- `IsNull()`. Creates an 'IsNotNull' expression. (Not supported)
- `StringMultiValuePropertyExpression(String)`. Constructor.

Remarks

- `FAST Search for Sharepoint 2010`. Multi-value properties are not supported when your search provider is Microsoft FAST Search for SharePoint 2010.
- `Search Server 2010`. Multi-value properties are not supported when your search provider is Microsoft Search Server 2010.
- `Solr`. Field names in Apache Solr are case sensitive. When constructing a `PropertyExpression` for a custom Solr index field, ensure that the proper casing is applied.

StringPropertyExpression

8.50 and higher

Defines an expression representing a string index field.

Namespace

```
Ektron.Cms.Search.Expressions.StringPropertyExpression
```

Properties

- `Category`. Gets or sets the CMS category by which this property is classified.
- `Name`. Gets or sets the internal property name.

- `ValueType`. Gets or sets the data type of the values that may be associated with this property.

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `AsMultiValue()`. Returns a reference to the multi-value form of this property expression, typically used to retrieve a collection of values from the index.

  NOTE: An index field must be specifically configured to process multi-value data for this to be applicable. Not all search providers support this behavior.

- `Contains(String)`. Constructs a `ContainsExpression` using the current object as its left branch and the specified phrase as its right branch.
- `Contains(String,WordForms)`. Constructs a `ContainsExpression` using the current object as its left branch and the specified phrase as its right branch.
- `EqualTo(String)`. Constructs an `EqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `GreaterThan(String)`. Constructs a `GreaterThanExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `GreaterThanOrEqualTo(String)`. Constructs a `GreaterThanOrEqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `LessThan(String)`. Constructs a `LessThanExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `LessThanOrEqualTo(String)`. Constructs a `LessThanOrEqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `NotEqualTo(String)`. Constructs a `NotEqualsExpression` using the current object as its left-branch, and the given parameter "value" as its right-branch.
- `StringPropertyExpression(String)`. Constructor.

Remarks

- `Solr`. Field names in Apache Solr are case sensitive. When constructing a `PropertyExpression` for a custom Solr index field, ensure that the proper casing is applied.

StringValueExpression

8.50 and higher

Defines an expression representing a string value.

Namespace

```
Ektron.Cms.Search.Expressions.StringValueExpression
```

Methods

- `Accept(ExpressionVisitor)`. Accepts a visitor to perform actions on this expression node.
- `StringValueExpression(String)`. Constructor.

Search Abstract Classes and Interfaces

ExpressionVisitor

8.60 and higher

The ExpressionVisitor class is an abstract base class defining objects that visit expression trees.

Namespace

```
Ektron.Cms.Search.ExpressionVisitor
```

Methods

- `Visit(AndExpression)`. Visits the specified AndExpression.
- `Visit(BooleanMultiValuePropertyExpression)`. Visits the specified BooleanMultiValuePropertyExpression.
- `Visit(BooleanPropertyExpression)`. Visits the specified BooleanPropertyExpression.
- `Visit(BooleanValueExpression)`. Visits the specified BooleanValueExpression.
- `Visit(ContainsExpression)`. Visits the specified ContainsExpression.
- `Visit(DateMultiValuePropertyExpression)`. Visits the specified DateMultiValuePropertyExpression.
- `Visit(DatePropertyExpression)`. Visits the specified DatePropertyExpression.
- `Visit(DateValueExpression)`. Visits the specified DateValueExpression.
- `Visit(DecimalMultiValuePropertyExpression)`. Visits the specified DecimalMultiValuePropertyExpression.
- `Visit(DecimalPropertyExpression)`. Visits the specified DecimalPropertyExpression.
- `Visit(DecimalValueExpression)`. Visits the specified DecimalValueExpression.

- `Visit(DefaultScopeExpression)`. Visits the specified DefaultScopeExpression.
- `Visit(EqualsExpression)`. Visits the specified EqualsExpression.
- `Visit(Expression)`. Visits the specified Expression.
- `Visit(GreaterThanExpression)`. Visits the specified GreaterThanExpression.
- `Visit(GreaterThanOrEqualsExpression)`. Visits the specified GreaterThanOrEqualsExpression.
- `Visit(IntegerMultiValuePropertyExpression)`. Visits the specified IntegerMultiValuePropertyExpression.
- `Visit(IntegerPropertyExpression)`. Visits the specified IntegerPropertyExpression.
- `Visit(IntegerValueExpression)`. Visits the specified IntegerValueExpression.
- `Visit(IsNotNullExpression)`. Visits the specified IsNotNullExpression.
- `Visit(IsNullExpression)`. Visits the specified IsNullExpression.
- `Visit(KeywordExpression)`. Visits the specified KeywordExpression.
- `Visit(LessThanExpression)`. Visits the specified LessThanExpression.
- `Visit(LessThanOrEqualsExpression)`. Visits the specified LessThanOrEqualsExpression.
- `Visit(NotEqualsExpression)`. Visits the specified NotEqualsExpression.
- `Visit(NotExpression)`. Visits the specified NotExpression.
- `Visit(OrExpression)`. Visits the specified OrExpression.
- `Visit(PropertyExpression)`. Visits the specified PropertyExpression.
- `Visit(QuotedStringValueExpression)`. Visits the specified QuotedStringValueExpression.
- `Visit(ScopeExpression)`. Visits the specified ScopeExpression.
- `Visit(StringMultiValuePropertyExpression)`. Visits the specified StringMultiValuePropertyExpression.
- `Visit(StringPropertyExpression)`. Visits the specified StringPropertyExpression.
- `Visit(StringValueExpression)`. Visits the specified StringValueExpression.

IAuthenticationHandler

8.50 and higher

The IAuthenticationHandler interface describes a component capable of applying the appropriate permission data to an outgoing search query.

Namespace

```
Ektron.Cms.Search.IAuthenticationHandler
```

Methods

- `Handle(AdministratorPermission)`. Handles the specified permission data, applying it to an outgoing search query. The permission data indicates that the query should execute as a CMS administrator.
- `Handle(CurrentUserPermission)`. Handles the specified permission data, applying it to an outgoing search query. The permission data indicates that the query should execute as the current CMS user.
- `Handle(ManualUserPermission)`. Handles the specified permission data, applying it to an outgoing search query. The permission data indicates that the query should execute as a specific CMS user.
- `Handle(Permission)`. Handles the specified permission data, applying it to an outgoing search query.

ICrawler

8.50 and higher

The ICrawler interface describes a components capable of managing initiation of indexing activities for relevant CMS events.

Namespace

```
Ektron.Cms.Search.ICrawler
```

Methods

- `AddAlias(AliasData)`. Signals the crawler to handle an add manual alias event.
- `AddAliasRule(AliasRuleData)`. Signals the crawler to handle an add auto alias event.
- `AddAutoAlias(UrlAliasAutoData)`. Signals the crawler to handle an add alias event.
- `AddBlogComment(BlogComment)`. Signals the crawler to handle an add blog comment event.
- `AddColleague(Int64,Int64)`. Signals the crawler to handle an add colleague event.
- `AddCommunityAlias(UrlAliasCommunityData)`. Signals the crawler to handle an add alias event.
- `AddContent(ContentData)`. Signals the crawler to handle an add content event.
- `AddCustomProperty(CustomPropertyData)`. Signals the crawler to handle a custom property add event.
- `AddCustomPropertyObject(CustomPropertyObjectData)`. Signals the crawler to handle a custom property object add event.

- `AddDiscussionReply(TaskData)`. Signals the crawler to handle an add discussion reply event.
- `AddDiscussionTopic(DiscussionTopic)`. Signals the crawler to handle an add discussion topic event.
- `AddFolder(FolderData)`. Signals the crawler to handle an add folder event.
- `AddGroup(CommunityGroupData)`. Signals the crawler to handle an add Group event.
- `AddLibraryItem(LibraryData)`. Signals the crawler to handle an add library item event.
- `AddManualAlias(UrlAliasManualData)`. Signals the crawler to handle an add alias event.
- `AddPermission(PermissionData)`. Signals the crawler to handle the addition of a permission.
- `AddProperty(MetaTypeBaseData)`. Signals the crawler to handle an add metadata event.
- `AddRegExAlias(UrlAliasRegExData)`. Signals the crawler to handle an add alias event.
- `AddTaxonomyItem(TaxonomyBaseData)`. Signals the crawler to handle an add taxonomy item event.
- `AddTemplate(TemplateData)`. Signals the crawler to handle the addition of a template.
- `AddUser(UserData)`. Signals the crawler to handle an add user event.
- `AddUserProperty(UserCustomPropertyData)`. Signals the crawler to handle a user property add event.
- `AddXmlConfiguration(XmlConfigData)`. Signals the crawler to handle an add XML configuration event.
- `CopyFolder(Int64,Int64)`. Signals the crawler to handle a copy folder event.
- `DeleteAlias(AliasData)`. Signals the crawler to handle a delete manual alias event.
- `DeleteAliasRule(AliasRuleData)`. Signals the crawler to handle a delete auto alias event.
- `DeleteAutoAlias(Int64)`. Signals the crawler to handle an delete alias event.
- `DeleteCatalogEntry(Int64)`. Signals the crawler to handle a delete catalog entry event.
- `DeleteColleague(Int64,Int64)`. Signals the crawler to handle an delete colleague event.
- `DeleteCommunityAlias(Int64)`. Signals the crawler to handle an delete alias event.
- `DeleteContent(Int64)`. Signals the crawler to handle an delete content event.
- `DeleteCustomProperty(Int64,Int32)`. Signals the crawler to handle a custom property delete event.

- `DeleteCustomPropertyObject (Int64,Int32,CustomPropertyObjectType,Int64)`. Signals the crawler to handle a custom property object delete event.
- `DeleteFolder(Int64)`. Signals the crawler to handle an delete folder event.
- `DeleteGroup(Int64)`. Signals the crawler to handle a delete group event.
- `DeleteLibraryItem(Int64)`. Signals the crawler to handle a delete library item event.
- `DeleteManualAlias(Int64)`. Signals the crawler to handle an delete alias event.
- `DeleteObjectTag(Int64,Int64,CMSObjectTypes,Int64)`. Signals the crawler to handle a tag delete event.
- `DeletePermission(PermissionData)`. Signals the crawler to handle the deletion of a permission.
- `DeleteProperty(Int64)`. Signals the crawler to handle an delete metadata event.
- `DeleteRegExAlias(Int64)`. Signals the crawler to handle an delete alias event.
- `DeleteTaxonomyItem(Int64)`. Signals the crawler to handle an delete taxonomy item event.
- `DeleteTemplate(Int64)`. Signals the crawler to handle the deletion of a template.
- `DeleteUser(Int64)`. Signals the crawler to handle a delete user event.
- `DeleteUserProperty(Int64)`. Signals the crawler to handle a user property delete event.
- `DeleteWebEvent(Int64)`. Signals the crawler to handle an delete Web event.
- `DeleteXmlConfiguration(Int64)`. Signals the crawler to handle an delete XML configuration event.
- `DisableAlias(AliasData)`. Signals the crawler to handle a disable manual alias event.
- `DisableAliasRule(AliasRuleData)`. Signals the crawler to handle a disable auto alias event.
- `EnableAlias(AliasData)`. Signals the crawler to handle an enable alias event.
- `EnableAliasRule(AliasRuleData)`. Signals the crawler to handle a enable auto alias event.
- `MoveFolder(Int64)`. Signals the crawler to handle a move folder event.
- `PublishCatalogEntry(Int64)`. Signals the crawler to handle a publish catalog entry event.
- `PublishWebEvent(WebEventData)`. Signals the crawler to handle an publish Web event.
- `StartFullCrawl()`. Requests a full crawl from the associated indexing service.
- `StartFullCrawl(CrawlType)`. Requests a full crawl from the associated indexing service.

- `StartFullCrawl(CrawlType,Boolean)`. Requests a full crawl from the associated indexing service with an option whether or not to respect crawl filters from CMS database.
- `StartIncrementalCrawl()`. Requests an incremental crawl from the associated indexing service.
- `StartIncrementalCrawl(Boolean)`. Requests an incremental crawl from the associated indexing service.
- `StartIncrementalCrawl(Boolean,CrawlType)`. Requests an incremental crawl from the associated indexing service.
- `StartIncrementalCrawl(Boolean,CrawlType,Boolean)`. Requests an incremental crawl from the associated indexing service with an option whether or not to respect crawl filters from CMS database.
- `StartIncrementalCrawl(CrawlType)`. Requests an incremental crawl from the associated indexing service.
- `TagObject(TagAssignmentData)`. Signals the crawler to handle a tag assignment event.
- `UpdateAlias(AliasData)`. Signals the crawler to handle an update manual alias event.
- `UpdateAliasRule(AliasRuleData)`. Signals the crawler to handle an update auto alias event.
- `UpdateAliasSetting(String,Object)`. Signals the crawler to handle a URL aliasing settings update.
- `UpdateAutoAlias(UrlAliasAutoData)`. Signals the crawler to handle an update alias event.
- `UpdateCommunityAlias(UrlAliasCommunityData)`. Signals the crawler to handle an update alias event.
- `UpdateContent(ContentData)`. Signals the crawler to handle an update content event.
- `UpdateCustomProperty(CustomPropertyData)`. Signals the crawler to handle a custom property update event.
- `UpdateCustomPropertyObject(CustomPropertyObjectData)`. Signals the crawler to handle a custom property object update event.
- `UpdateDiscussionTopic(DiscussionTopic)`. Signals the crawler to handle an update discussion topic event.
- `UpdateFolder(FolderData)`. Signals the crawler to handle an update folder event.
- `UpdateGroup(CommunityGroupData)`. Signals the crawler to handle an update group event.
- `UpdateItemInheritance(Int64,String,Boolean)`. Signals the crawler to handle the update of the permission inheritance setting for a particular item.
- `UpdateLibraryItem(LibraryData)`. Signals the crawler to handle an update library item event.

- `UpdateManualAlias(UrlAliasManualData)`. Signals the crawler to handle an update alias event.
- `UpdatePermission(PermissionData)`. Signals the crawler to handle the update of a permission.
- `UpdatePrivateSetting(Int64,String,Boolean)`. Signals the crawler to handle the update of the permission privacy setting for a particular item.
- `UpdateProperty(MetaTypeBaseData)`. Signals the crawler to handle an update metadata event.
- `UpdateRegExAlias(UrlAliasRegExData)`. Signals the crawler to handle an update alias event.
- `UpdateTaxonomyItem(TaxonomyBaseData)`. Signals the crawler to handle an update taxonomy item event.
- `UpdateTemplate(TemplateData)`. Signals the crawler to handle the update of a template.
- `UpdateUser(UserData)`. Signals the crawler to handle an update user event.
- `UpdateUserProperty(UserCustomPropertyData)`. Signals the crawler to handle a user property update event.
- `UpdateXmlConfiguration(XmlConfigData)`. Signals the crawler to handle an update XML configuration event.
- `UpdateXmlIndex(XmlConfigData)`. Signals the crawler to handle an update to XML Index configuration event (saving the index fields on second screen of Smart Form creation).

IIntegratedSearchMapping

8.50 and higher

The IntegratedSearchMapping interface describes a business object which exposes functionality supporting the creation and management of integrated search mappings.

Namespace

```
Ektron.Cms.Search.IIntegratedSearchMapping
```

Methods

- `Add(IntegratedSearchMappingData)`. Adds the specified integrated search mapping.
- `Delete(Guid)`. Deletes the specified integrated search mapping from the database.
- `Get()`. Gets a collection of all integrated search mappings.
- `Get(Guid)`. Gets the specified integrated search mapping.
- `Update(IntegratedSearchMappingData)`. Updates the specified integrated search mapping.

IPropertyNameResolver

8.50 and higher

The IPropertyNameResolver interface describes a component capable of transforming an internal CMS property name to the name of the corresponding index field.

Namespace

```
Ektron.Cms.Search.IPropertyNameResolver
```

Methods

- `Resolve(String,PropertyCategory,PropertyType)`. Translates the specified CMS property name to the name of the corresponding index field (as its known the underlying search provider).

ISearchManager

8.50 and higher

The ISearchManager interface defines a component capable of submitting queries for CMS data (content, users, groups, and so on) and returning the relevant results.

Namespace

```
Ektron.Cms.Search.ISearchManager
```

Methods

- `Search(AdvancedSearchCriteria)`. Submits a query according to the specified criteria.
- `Search(KeywordSearchCriteria)`. Submits a query according to the specified criteria.

ISearchProvider

8.50 and higher

The ISearchProvider interface describes a component that is capable of communicating with a specific search engine. This includes the translation of CMS search criteria into a syntax supported by that engine, issuing of the query, and the processing of any response data.

Namespace

```
Ektron.Cms.Search.ISearchProvider
```

Methods

- `Search(AdvancedSearchCriteria)`. Executes a narrow, expression-based query using the specified criteria.
- `Search(KeywordSearchCriteria)`. Executes a keyword-based query using the specified criteria.

ISearchSettings

8.50 and higher

The ISearchSettings interface describes a component capable of updating and retrieving the configuration details for the search feature.

Namespace

```
Ektron.Cms.Search.ISearchSettings
```

Methods

- `GetItem()`. Gets the search settings for the current site.

ISuggestedResults

8.50 and higher

The ISuggestedResults interface describes a component providing administrative capabilities for suggested search result data.

Namespace

```
Ektron.Cms.Search.ISuggestedResults
```

Methods

- `Add(SuggestedResultSet)`. Adds the specified suggested result set.
- `Delete(Guid)`. Deletes the suggested result identified by the ID.
- `GetItem(Guid)`. Gets the suggested result set identified by the specified ID.
- `GetItem(String)`. Gets the suggested result set containing the specified search phrase.
- `GetList()`. Gets a collection of suggested result sets from the configured search provider.
- `Update(SuggestedResultSet)`. Updates the specified suggested result set.

ISynonyms

8.50 and higher

The ISynonyms interface describes a class providing access to search synonym sets.

Namespace

```
Ektron.Cms.Search.ISynonyms
```

Methods

- `Add(SynonymSet)`. Adds the specified synonym set.
- `Add(List<SynonymSet>)`. Adds the specified synonym sets.
- `Delete(Guid,Int32)`. Deletes the specified synonym set.
- `GetItem(Guid)`. Gets a synonym set.
- `GetList(Int32)`. Gets a collection of synonym sets for the specified language.
- `Update(List<SynonymSet>)`. Updates the specified synonym sets.
- `Update(SynonymSet)`. Updates the specified synonym set.

Settings

8.50 and higher

The Settings manager category manages CMS settings and has the following classes:

- *CmsMessageManager* on page 1424. Manages messages in the Ektron CMS.
- *CmsMessageTypeManager* on page 1442. Manages message types in the Ektron CMS.
- *CustomPropertyManager* on page 1456. Manages user-specified properties to create custom fields.
- *DXH* on page 1469. Sub-category of Settings. 9.00 and higher
  - *DXHUserConnectionManager* on page 1469. Manages connections between the Digital Experience Hub (DXH) and other applications.
- *Notifications* on page 1479. Sub-category of Settings.
  - *NotificationAgentSettingManager* on page 1480. Manages notification agents.
  - *NotificationPreferenceManager* on page 1498. Manages preferences for notification agents.
  - *NotificationPublishPreferenceManager* on page 1523. Manages preferences for notification agents that can be published.
  - *UserNotificationSettingManager* on page 1535. Manages user-specified notifications.
- *PermissionManager* on page 1552. Manages user permissions.
- *SmartformConfigurationManager* on page 1577. Manages Smart Form configurations.
- *TaskCategoryManager* on page 1600. Manages task categories.
- *TaskCommentManager* on page 1593. Manages task comments.
- *TaskManager* on page 1612. Manages tasks.
- *TaxonomyCustomPropertyManager* on page 1631. Manages user-specified taxonomy properties.
- *UrlAliasing* on page 1649. Sub-category of Settings.
  - *AliasManager* on page 1650. Manages aliases.
  - *AliasRuleManager* on page 1673. Manages alias rules.
  - *AliasSettingManager* on page 1704. Manages alias settings.
  - *AutoAliasManager* on page 1736. Manages automatic aliases, which lets you assign an alias to several content items at once.
  - *CommonAliasManager* on page 1755. Gets alias and target data that is common to any alias.

- *CommunityAliasManager* on page 1759. Manages aliases that apply to communities.
- *ManualAliasManager* on page 1776. Manages manual aliases.
- *RedirectManager* on page 1792. Manages URL redirect addresses.
- *RegExAliasManager* on page 1812. Manages regular expressions that replace URLs that cannot be read.

CmsMessageManager

8.50 and higher

The CmsMessageManager class manages messages in the Ektron CMS.

Namespace

```
Ektron.Cms.Framework.Settings.CmsMessageManager
```

Constructors

- `CmsMessageManager()`
- `CmsMessageManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `MessageService`. Gets instance to ICmsMessage.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 1427
- *GetDefaultItemByType* on page 1428
- *GetItem* on page 1430
- *GetItemCollection* on page 1432
- *Update* on page 1437

Add

```
Add(Ektron.Cms.Messaging.CmsMessageData)
```

Adds a message based on information in an CmsMessageData object. The method populates the message.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- Is Default
- * Message Type ID
- * Subject
- * HTML Body

Parameters

- `message`. The CmsMessageData object to add.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-3
last" runat="server" Text="* Title:" />
        <ektronUI:TextField  ID="uxTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsDefaultLabel" AssociatedControlID="uxIsDefault"
CssClass="span-3 last" runat="server" Text=" Is Default:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsDefault" CssClass="span-
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageTypeIdLabel" AssociatedControlID="uxMessageTypeId"
CssClass="span-3 last" runat="server" Text="* Message Type Id:" />
        <ektronUI:TextField  ID="uxMessageTypeId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="5" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSubjectLabel" AssociatedControlID="uxSubject"
CssClass="span-3 last" runat="server" Text="* Subject:" />
        <ektronUI:TextField  ID="uxSubject" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxHtmlBodyLabel" AssociatedControlID="uxHtmlBody"
CssClass="span-3 last" runat="server" Text="* Html Body:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxHtmlBody" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Messaging;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CmsMessageTypeManager cmsMessageTypeManager = new CmsMessageTypeManager();
            CmsMessageTypeData cmsMessageTypeData = cmsMessageTypeManager.GetItem
(long.Parse(uxMessageTypeId.Text));

            CmsMessageManager cmsMessageManager = new CmsMessageManager();
            CmsMessageData cmsMessageData = new CmsMessageData()
            {
                Title = uxTitle.Text ,
                IsDefaultMessage = uxIsDefault.Checked,
                Subject = uxSubject.Text ,
                MessageType = cmsMessageTypeData,
                HtmlBody = uxHtmlBody.Text
            };

            cmsMessageManager.Add(cmsMessageData);

            MessageUtilities.UpdateMessage(uxMessage,  "Message data added with Id " +
cmsMessageData.Id, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
```

```
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a message (CmsMessageData object) from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The ID of the message to delete

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageIdLabel" AssociatedControlID="uxMessageId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxMessageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Messaging;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
             long messageId = long.Parse(uxMessageId.Text);
             CmsMessageManager cmsMessageManager = new CmsMessageManager();
             CmsMessageData cmsMessageData = cmsMessageManager.GetItem(messageId);

             if (cmsMessageData != null)
             {
                 cmsMessageManager.Delete(messageId);
                 MessageUtilities.UpdateMessage(uxMessage, "Message deleted.",
Message.DisplayModes.Success);
             }
             else
             {
                 MessageUtilities.UpdateMessage(uxMessage, "Message does not exists.",
Message.DisplayModes.Error);
             }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetDefaultItemByType

```
GetDefaultItemByType(System.Int64)
```

Retrieves the default message for a message type in the CMS. All available languages of the message are returned as part of the CmsMessageData object collection.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `messageTypeId`. The ID of the message type to retrieve the default message for.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageTypeIdLabel" AssociatedControlID="uxMessageTypeId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxMessageTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxMessageType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSubject" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxHtmlBody" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Messaging;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
```

```
        {
            long messageTypeId = long.Parse(uxMessageTypeId.Text);

            CmsMessageManager cmsMessageManager = new CmsMessageManager();
            CmsMessageDataCollection cmsMessageData =
cmsMessageManager.GetDefaultItemByType(messageTypeId);

            if (cmsMessageData != null && cmsMessageData.Count > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for message with ID "
+ cmsMessageData[0].Id.ToString() + " are below (Only top item in the collection is
displayed)", Message.DisplayModes.Success);

                uxTitle.Text = "Title : " + cmsMessageData[0].Title;
                uxMessageType.Text = "Message Type : " + cmsMessageData
[0].MessageType.Name;
                uxSubject.Text = "Subject : " + cmsMessageData[0].Subject;
                uxHtmlBody.Text = "Html Body : " + cmsMessageData[0].HtmlBody;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Message does not exists.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.CmsMessageData)
```

Retrieves the properties of a specific CmsMessageData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The ID of the message to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageIdLabel" AssociatedControlID="uxMessageId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxMessageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxMessageType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSubject" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxHtmlBody" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Messaging;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageId = long.Parse(uxMessageId.Text);

            CmsMessageManager cmsMessageManager = new CmsMessageManager();
            CmsMessageData cmsMessageData = cmsMessageManager.GetItem(messageId);
```

```
            if (cmsMessageData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for message with ID "
+ cmsMessageData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxTitle.Text = "Title : " + cmsMessageData.Title;
                uxMessageType.Text = "Message Type : " +
cmsMessageData.MessageType.Name;
                uxSubject.Text = "Subject : " + cmsMessageData.Subject;
                uxHtmlBody.Text = "Html Body : " + cmsMessageData.HtmlBody;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Message does not exists.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItemCollection

```
GetItemCollection(System.Int64)
```

Retrieves a single message from the CMS in all available languages.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the message to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageIdLabel" AssociatedControlID="uxMessageId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxMessageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxMessageType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSubject" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxHtmlBody" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Messaging;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageId = long.Parse(uxMessageId.Text);

            CmsMessageManager cmsMessageManager = new CmsMessageManager();
            CmsMessageDataCollection cmsMessageData =
cmsMessageManager.GetItemCollection(messageId);

            if (cmsMessageData != null && cmsMessageData.Count > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for message with ID "
+ cmsMessageData[0].Id.ToString() + " are below (Only top item in the collection is
displayed)", Message.DisplayModes.Success);
```

```
                uxTitle.Text = "Title : " + cmsMessageData[0].Title;
                uxMessageType.Text = "Message Type : " + cmsMessageData
[0].MessageType.Name;
                uxSubject.Text = "Subject : " + cmsMessageData[0].Subject;
                uxHtmlBody.Text = "Html Body : " + cmsMessageData[0].HtmlBody;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Message does not exists.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(CmsMessageCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- CMS Message Property
- * Object Value

Parameters

- `criteria`. CmsMessageCriteria used to retrieve messages.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCmsMessagePropertyLabel"
AssociatedControlID="uxCmsMessageProperty" CssClass="span-6 last" runat="server"
Text="CmsMessageProperty:" />
        <asp:DropDownList ID="uxCmsMessageProperty" runat="server">
            <asp:ListItem>MessageTypeId</asp:ListItem>
            <asp:ListItem>Title</asp:ListItem>
        </asp:DropDownList>
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCmsMessageDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
                        Subject
                    </th>
                    <th>
                        Html Body
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Subject")%>
            </td>
            <td class="devsite-method">
```

```
                <%# Eval("HtmlBody")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Messaging;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            CmsMessageManager cmsMessageManager = new CmsMessageManager();

            CmsMessageCriteria criteria = new CmsMessageCriteria();
            if (uxCmsMessageProperty.SelectedItem.Text == "MessageTypeId")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(CmsMessageProperty.MessageTypeId ,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CmsMessageProperty.Title ,
CriteriaFilterOperator.Contains, Objectvalue);
            }
            criteria.AddFilter(CmsMessageProperty.LanguageId,
CriteriaFilterOperator.EqualTo, cmsMessageManager.RequestInformation.ContentLanguage);


            List<CmsMessageData> cmsMessageDataList = cmsMessageManager.GetList
(criteria);

            uxCmsMessageDataListView.Visible = true;
            uxCmsMessageDataListView.DataSource = cmsMessageDataList;
            uxCmsMessageDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
```

```
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.CmsMessageData)
```

Updates an existing CmsMessageData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- Title
- Message Type ID
- Subject
- HTML Body

Parameters

- `message`. The CmsMessageData object to update.

Returns

Returns the custom CmsData object updated.

Remarks

Validate the CmsMessage item with `GetItem()` before you update the properties. Then, call `Update` with your modified CmsMessageData object.

NOTE: You cannot change all properties after the initial Add event, such as the CmsMessageData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageIdLabel" AssociatedControlID="uxMessageId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField  ID="uxMessageId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-3
last" runat="server" Text=" Title:" />
```

```
        <ektronUI:TextField  ID="uxTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageTypeIdLabel" AssociatedControlID="uxMessageTypeId"
CssClass="span-3 last" runat="server" Text=" Message Type Id:" />
        <ektronUI:TextField  ID="uxMessageTypeId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSubjectLabel" AssociatedControlID="uxSubject"
CssClass="span-3 last" runat="server" Text=" Subject:" />
        <ektronUI:TextField  ID="uxSubject" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxHtmlBodyLabel" AssociatedControlID="uxHtmlBody"
CssClass="span-3 last" runat="server" Text=" Html Body:" />
        <asp:TextBox TextMode="MultiLine" Rows="3"  ID="uxHtmlBody" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Messaging;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageId = long.Parse(uxMessageId.Text);

            CmsMessageTypeManager cmsMessageTypeManager = new CmsMessageTypeManager();
            CmsMessageManager cmsMessageManager = new CmsMessageManager();
            CmsMessageData cmsMessageData = cmsMessageManager.GetItem(messageId);

            if (cmsMessageData != null)
            {
```

```
                cmsMessageData.Title = (uxTitle.Text != string.Empty ? uxTitle.Text :
cmsMessageData.Title);
                cmsMessageData.Subject = (uxSubject.Text != string.Empty ?
uxSubject.Text : cmsMessageData.Subject);
                cmsMessageData.HtmlBody = (uxHtmlBody.Text != string.Empty ?
uxHtmlBody.Text : cmsMessageData.HtmlBody);
                cmsMessageData.MessageType = (uxMessageTypeId.Text != string.Empty ?
cmsMessageTypeManager.GetItem(long.Parse(uxMessageTypeId.Text)) :
cmsMessageData.MessageType);

                cmsMessageManager.Update(cmsMessageData);

                MessageUtilities.UpdateMessage(uxMessage,  "Message Updated",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Message does not exists.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

CmsMessageCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Messaging
```

Constructors

- `CmsMessageCriteria()`

  ```
  public CmsMessageCriteria()
  ```

- `CmsMessageCriteria(Ektron.Cms.Messaging.CmsMessageProperty, EkEnumeration.OrderByDirection)`

  ```
  public CmsMessageCriteria(Ektron.Cms.Messaging.CmsMessageProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `CmsMessageProperty` are:

  - HtmlBody
  - Id
  - IsDefault

- LanguageId
- MessageTypeId
- MessageTypeName
- MessageTypeScope
- SiteId
- Subject
- TextBody
- Title

CmsMessageData

Namespace

```
Ektron.Cms.Messaging
```

Properties

- `CmsMessageData()`

```
public CmsMessageData()
```

- `CmsMessageData(string, string, string, string, int, long, bool)`

```
public CmsMessageData(string title, string subject,
  string htmlBody, string textBody, int languageId,
  long messageTypeId, bool isDefault)
```

- `HtmlBody`. Gets or sets the Html Body of the Message. Depending on the delivery mechanism of the message, either the TextBody or HtmlBody may be used.

```
public string HtmlBody { set; get; }
```

- `IsDefaultMessage`. Gets or sets the IsDefault flag for this message. If true, this message is the defualt used for the MessageType.

```
public bool IsDefaultMessage { set; get; }
```

- `MessageType`. Gets or sets the the message type of this message.

```
public Ektron.Cms.Messaging.CmsMessageTypeData
  MessageType { set; get; }
```

- `SiteId`. Gets or sets the Id of the site that the message is associated with.

```
public long SiteId { set; get; }
```

- `Subject`. Gets or sets the Subject of the Message.

```
public string Subject { set; get; }
```

- `TextBody`. Gets or sets the Html Body of the Message. Depending on the delivery mechanism of the message, either the TextBody or HtmlBody may be used.

```
public string TextBody { set; get; }
```

- `Title`. Gets or Sets the Title of the Message.

```
public string Title { set; get; }
```

- `Ektron.Cms.Messaging.CmsMessageData.Validate()`. Validates that the data object is in a valid state for persisting. It returns a collection of ValidationResult messages if the object is not valid.

```
public override ValidationResults Validate()
```

CmsMessageTypeManager

8.50 and higher

The CmsMessageTypeManager class manages message types in the Ektron CMS.

Namespace

```
Ektron.Cms.Framework.Settings.CmsMessageTypeManager
```

Constructors

- `CmsMessageTypeManager()`
- `CmsMessageTypeManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `MessageTypeService`. Gets instance to ICmsMessageType.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Messaging.CmsMessageTypeData)
```

Adds a message type based on information in a CmsMessageTypeData object. The method populates the message.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Name
- * Scope

Parameters

- `messageType`. The CmsMessageTypeData object to add.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-3
last" runat="server" Text="* Name:" />
        <ektronUI:TextField  ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxScopeLabel" AssociatedControlID="uxScope" CssClass="span-3
last" runat="server" Text="* Scope:" />
        <ektronUI:TextField  ID="uxScope" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms.Messaging;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CmsMessageTypeManager cmsMessageTypeManager = new CmsMessageTypeManager();
            CmsMessageTypeData cmsMessageTypeData = new CmsMessageTypeData()
            {
                 Name = uxName.Text ,
                 Scope = uxScope.Text
            };

            cmsMessageTypeManager.Add(cmsMessageTypeData);

            MessageUtilities.UpdateMessage(uxMessage, "Message Type added with Id " +
cmsMessageTypeData.Id, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a message type (CmsMessageTypeData object) from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the message to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageTypeIdLabel" AssociatedControlID="uxMessageTypeId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxMessageTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Messaging;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageTypeId = long.Parse(uxMessageTypeId.Text);

            CmsMessageTypeManager cmsMessageTypeManager = new CmsMessageTypeManager();
            CmsMessageTypeData cmsMessageTypeData = cmsMessageTypeManager.GetItem
(messageTypeId);

            if (cmsMessageTypeData != null)
            {
                cmsMessageTypeManager.Delete(messageTypeId);
                MessageUtilities.UpdateMessage(uxMessage, "Message Type deleted.",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Message Type does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
```

```
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.CmsMessageTypeData)
```

Retrieves the properties of a specific CmsMessageTypeData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the message type to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageTypeIdLabel" AssociatedControlID="uxMessageTypeId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxMessageTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxScope" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Messaging;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageTypeId = long.Parse(uxMessageTypeId.Text);

            CmsMessageTypeManager cmsMessageTypeManager = new CmsMessageTypeManager();
            CmsMessageTypeData cmsMessageTypeData = cmsMessageTypeManager.GetItem
(messageTypeId);

            if (cmsMessageTypeData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for message with ID "
+ cmsMessageTypeData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxScope.Text = "Scope : " + cmsMessageTypeData.Scope;
                uxName.Text = "Name : " + cmsMessageTypeData.Name;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Message Type does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(CmsMessageTypeCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- CMS Message Type Property
- * Object Value

Parameters

- `criteria`. CmsMessageTypeCriteria used to retrieve message types.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCmsMessageTypePropertyLabel"
AssociatedControlID="uxCmsMessageTypeProperty" CssClass="span-6 last" runat="server"
Text="CmsMessageTypeProperty:" />
        <asp:DropDownList ID="uxCmsMessageTypeProperty" runat="server">
            <asp:ListItem>Name</asp:ListItem>
            <asp:ListItem>Scope</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCmsMessageTypeDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Name
                    </th>
```

```
                    <th>
                        Scope
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Scope")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Messaging;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            CmsMessageTypeCriteria criteria = new CmsMessageTypeCriteria
(CmsMessageTypeProperty.Id, EkEnumeration.OrderByDirection.Descending);
            if (uxCmsMessageTypeProperty.SelectedItem.Text == "Name")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CmsMessageTypeProperty.Name ,
CriteriaFilterOperator.Contains , Objectvalue);
            }
            else
            {
```

```
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CmsMessageTypeProperty.Scope ,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            CmsMessageTypeManager cmsMessageTypeManager = new CmsMessageTypeManager();
            List<CmsMessageTypeData> cmsMessageDataList = cmsMessageTypeManager.GetList
(criteria);

            uxCmsMessageTypeDataListView.Visible = true;
            uxCmsMessageTypeDataListView.DataSource = cmsMessageDataList;
            uxCmsMessageTypeDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTokenList

```
GetTokenList(System.Int64)
```

Retrieves the list of applicable replacement tokens for a message type ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `messageTypeId`. ID of the message type to retrieve tokens for.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageTypeIdLabel" AssociatedControlID="uxMessageTypeId"
CssClass="span-6 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxMessageTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Token List"></ektronUI:Button>
```

```
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxTokenListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Token
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Container.DataItem %>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageTypeId = long.Parse(uxMessageTypeId.Text);

            CmsMessageTypeManager cmsMessageTypeManager = new CmsMessageTypeManager();
            List<string> TokenList = cmsMessageTypeManager.GetTokenList(messageTypeId);

            uxTokenListView.Visible = true;
```

```
            uxTokenListView.DataSource = TokenList;
            uxTokenListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.CmsMessageTypeData)
```

Updates an existing CmsMessageTypeData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- Name
- Scope

Parameters

- `messageType`. The CmsMessageTypeData object to update.

Returns

Returns the custom CmsData object updated.

Remarks

Validate the CmsMessageType item with `GetItem()` before you update the properties. Then, call `Update` with your modified CmsMessageTypeData object.

NOTE: You cannot change all properties after the initial Add event, such as the CmsMessageTypeData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxMessageTypeIdLabel" AssociatedControlID="uxMessageTypeId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxMessageTypeId" CssClass="span-6" runat="server"
```

```
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-3
last" runat="server" Text=" Name:" />
        <ektronUI:TextField  ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxScopeLabel" AssociatedControlID="uxScope" CssClass="span-3
last" runat="server" Text=" Scope:" />
        <ektronUI:TextField  ID="uxScope" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Messaging;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long messageTypeId = long.Parse(uxMessageTypeId.Text);

            CmsMessageTypeManager cmsMessageTypeManager = new CmsMessageTypeManager();
            CmsMessageTypeData cmsMessageTypeData = cmsMessageTypeManager.GetItem
(messageTypeId);

            if (cmsMessageTypeData != null)
            {
                cmsMessageTypeData.Scope = uxScope.Text != string.Empty ? uxScope.Text :
cmsMessageTypeData.Scope;
                cmsMessageTypeData.Name = uxName.Text != string.Empty ? uxName.Text :
cmsMessageTypeData.Name;

                cmsMessageTypeManager.Update(cmsMessageTypeData);
```

```
                MessageUtilities.UpdateMessage(uxMessage,  "Message Type updated.",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Message Type does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

CmsMessageTypeCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Messaging
```

Constructors

- `CmsMessageTypeCriteria()`

  ```
  public CmsMessageTypeCriteria()
  ```

- `CmsMessageTypeCriteria (Ektron.Cms.Messaging.CmsMessageTypeProperty, EkEnumeration.OrderByDirection)`

  ```
  public CmsMessageTypeCriteria(Ektron.Cms.Messaging.CmsMessageTypeProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `CmsMessageTypeProperty` are:

  - Id
  - Name
  - Scope

CmsMessageTypeData

Namespace

```
Ektron.Cms.Messaging
```

Properties

- `Id`

```
public long Id { set; get; }
```

- `Name`. Gets or sets the name of the message type.

```
public string Name { set; get; }
```

- `Scope`. Gets or sets the scope of the message type.

```
public string Scope { set; get; }
```

- `Validate()`. Validates that the data object is in a valid state for persisting and returns a collection of ValidationResult messages if the object is not valid.

```
public override ValidationResults Validate()
```

CustomPropertyManager

8.50 and higher

The CustomPropertyManager class manages user-specified properties to create custom fields. See Managing Users and User Groups in the Ektron Reference for information about custom user properties.

Namespace

```
Ektron.Cms.Framework.Settings.CustomPropertyManager
```

Constructors

- `CustomPropertyManager()`
- `CustomPropertyManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `CustomPropertyManagerService`. Returns an instance of the business objects CustomPropertyManager service.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Messaging.UserCustomPropertyData)
```

Adds a UserCustomProperty, with details from the supplied UserCustomPropertyData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Name
- * Input Type
- * Type
- * Validation Type
- Message

Parameters

- `UCPData`

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-3
last" runat="server" Text="* Name:" />
        <ektronUI:TextField  ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxInputTypeLabel" AssociatedControlID="uxInputType"
CssClass="span-3 last" runat="server" Text="* Input Type:" />
        <asp:DropDownList ID="uxInputType" runat="server">
            <asp:ListItem>TextBox</asp:ListItem>
            <asp:ListItem>TextArea</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTypeLabel" AssociatedControlID="uxType" CssClass="span-3
last" runat="server" Text="* Type:" />
        <asp:DropDownList ID="uxType" runat="server">
            <asp:ListItem>String</asp:ListItem>
            <asp:ListItem>Numeric</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxValidationTypeLabel"
```

```
AssociatedControlID="uxValidationType" CssClass="span-3 last" runat="server" Text="*
Validation Type:" />
        <asp:DropDownList ID="uxValidationType" runat="server">
            <asp:ListItem>No Validation</asp:ListItem>
            <asp:ListItem>Cannot Be Blank</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxValMessageLabel" AssociatedControlID="uxValMessage"
CssClass="span-3 last" runat="server" Text=" Message:" />
        <ektronUI:TextField  ID="uxValMessage" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CustomPropertyManager customPropertyManager = new CustomPropertyManager();

            UserCustomPropertyData userCustomPropertyData = new UserCustomPropertyData()
            {
                  Name = uxName.Text ,
                  PropertyDisplayValueType = (uxInputType.SelectedItem.Value ==
"TextBox" ? Ektron.Cms.Common.EkEnumeration.ObjectPropertyDisplayTypes.TextBox :
Ektron.Cms.Common.EkEnumeration.ObjectPropertyDisplayTypes.TextArea),
                  PropertyValueType = (uxType.SelectedItem.Value =="String"?
Ektron.Cms.Common.EkEnumeration.ObjectPropertyValueTypes.String :
Ektron.Cms.Common.EkEnumeration.ObjectPropertyValueTypes.Numeric),
                  PropertyValidationMessage = uxValMessage.Text ,
                  PropertyValidationType = (uxValidationType.SelectedItem.Value == "No
Validation" ? 0 : 1)
            };
```

```
            customPropertyManager.Add(userCustomPropertyData);

            MessageUtilities.UpdateMessage(uxMessage, "User custom property added with
Id " + userCustomPropertyData.Id, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a UserCustomPropertyData object from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The identifier of the UserCustomPropertyData object to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCustomUserPropertyIdLabel"
AssociatedControlID="uxCustomUserPropertyId" CssClass="span-4 last" runat="server"
Text="* Id:" />
        <ektronUI:TextField ID="uxCustomUserPropertyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long customUserPropertyId = long.Parse(uxCustomUserPropertyId.Text);

            CustomPropertyManager customPropertyManager = new CustomPropertyManager();
            UserCustomPropertyData userCustomPropertyData =
customPropertyManager.GetItem(customUserPropertyId);

            if (userCustomPropertyData != null)
            {
                customPropertyManager.Delete(customUserPropertyId);
                MessageUtilities.UpdateMessage(uxMessage, "Custom property deleted.",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Custom property does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.UserCustomPropertyData)
```

Retrieves the properties of a specific UserCustomPropertyData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the UserCustomProperty to get.

Returns

UserCustomProperty details in a UserCustomPropertyData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserCustomPropertyIdLabel"
AssociatedControlID="uxUserCustomPropertyId" CssClass="span-3 last" runat="server"
Text="* Id :" />
        <ektronUI:TextField ID="uxUserCustomPropertyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDisplayValueType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxValueType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxValidationType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLanguageId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long userCustomPropertyId = long.Parse(uxUserCustomPropertyId.Text);

            CustomPropertyManager customPropertyManager = new CustomPropertyManager();
            UserCustomPropertyData userCustomPropertyData =
customPropertyManager.GetItem(userCustomPropertyId);

            if (userCustomPropertyData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for user custom
property with ID " + userCustomPropertyData.Id.ToString() + " are below",
Message.DisplayModes.Success);

                uxName.Text = "Name : " + userCustomPropertyData.Name;
                uxDisplayValueType.Text = "Display Value Type : " +
userCustomPropertyData.PropertyDisplayValueType;
                uxValueType.Text = "Value Type : " +
userCustomPropertyData.PropertyValueType;
                uxValidationType.Text = "Validation Type : " +
userCustomPropertyData.PropertyValidationType;
                uxLanguageId.Text = "Language Id : " +
userCustomPropertyData.ContentLanguage;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "User custom property does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(CustomPropertyCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- User Customer Property
- * Object Value

Parameters

- `criteria`. CustomPropertyCriteria Used to specify, or filter, the UserCustomPropertyData object to return.

Returns

A list of UserCustomPropertyData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserCustomPropertyLabel"
AssociatedControlID="uxUserCustomProperty" CssClass="span-6 last" runat="server"
Text="UserCustomProperty:" />
        <asp:DropDownList ID="uxUserCustomProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxUserCustomPropertyDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
```

```
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Name
                    </th>
                    <th>
                        Property Display Value Type
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("PropertyDisplayValueType")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.User;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            object Objectvalue;

            CustomPropertyCriteria criteria = new CustomPropertyCriteria();
            if (uxUserCustomProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = int.Parse(uxObjectValue.Text);
                criteria.AddFilter(UserCustomProperty.Id ,
CriteriaFilterOperator.EqualTo  , Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(UserCustomProperty.Name,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            CustomPropertyManager customPropertyManager = new CustomPropertyManager();
            List<UserCustomPropertyData> userCustomPropertyDataList =
customPropertyManager.GetList(criteria);

            uxUserCustomPropertyDataListView.Visible = true;
            uxUserCustomPropertyDataListView.DataSource = userCustomPropertyDataList;
            uxUserCustomPropertyDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.UserCustomPropertyData)
```

Updates a UserCustomProperty, with details from the supplied UserCustomPropertyData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- Name
- Input Type

- Validation Type
- Message

Parameters

- `UCPData`

Returns

Returns the custom CmsData object updated.

Remarks

Validate the object with `GetItem()` before you update the properties. Then, call `Update` with your modified UserCustomPropertyData object.

NOTE: You cannot change all properties after the initial Add event, such as the UserPermissionData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserCustomPropertyIdLabel"
AssociatedControlID="uxUserCustomPropertyId" CssClass="span-3 last" runat="server"
Text="* Id:" />
        <ektronUI:TextField  ID="uxUserCustomPropertyId" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-3
last" runat="server" Text=" Name:" />
        <ektronUI:TextField  ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxInputTypeLabel" AssociatedControlID="uxInputType"
CssClass="span-3 last" runat="server" Text=" Input Type:" />
        <asp:DropDownList ID="uxInputType" runat="server">
            <asp:ListItem>TextBox</asp:ListItem>
            <asp:ListItem>TextArea</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxValidationTypeLabel"
AssociatedControlID="uxValidationType" CssClass="span-3 last" runat="server" Text="
Validation Type:" />
        <asp:DropDownList ID="uxValidationType" runat="server">
            <asp:ListItem>No Validation</asp:ListItem>
            <asp:ListItem>Cannot Be Blank</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxValMessageLabel" AssociatedControlID="uxValMessage"
CssClass="span-3 last" runat="server" Text=" Message:" />
        <ektronUI:TextField  ID="uxValMessage" CssClass="span-4" runat="server"
```

```
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long userCustomPropertyId = long.Parse(uxUserCustomPropertyId.Text);

            CustomPropertyManager customPropertyManager = new CustomPropertyManager();
            UserCustomPropertyData userCustomPropertyData =
customPropertyManager.GetItem(userCustomPropertyId);

            if (userCustomPropertyData != null)
            {
                userCustomPropertyData.Name = (uxName.Text != string.Empty ? uxName.Text
: userCustomPropertyData.Name);
                userCustomPropertyData.PropertyDisplayValueType =
(uxInputType.SelectedItem.Value == "TextBox" ?
Ektron.Cms.Common.EkEnumeration.ObjectPropertyDisplayTypes.TextBox :
Ektron.Cms.Common.EkEnumeration.ObjectPropertyDisplayTypes.TextArea);
                userCustomPropertyData.PropertyValidationMessage = (uxValMessage.Text !=
string.Empty ? uxValMessage.Text :   userCustomPropertyData.PropertyValidationMessage);
                userCustomPropertyData.PropertyValidationType =
(uxValidationType.SelectedItem.Value == "No Validation" ? 0 : 1);
                customPropertyManager.Update(userCustomPropertyData);

                MessageUtilities.UpdateMessage(uxMessage, "User custom property
updated", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "User custom property does not
exists.", Message.DisplayModes.Error);
```

```
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

CustomPropertyCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.User
```

Constructors

- `CustomPropertyCriteria()`

  ```
  public CustomPropertyCriteria()
  ```

- `CustomPropertyCriteria (Ektron.Cms.Framework.Settings.CustomPropertyProperty, EkEnumeration.OrderByDirection)`

  ```
  public CustomPropertyCriteria(Ektron.Cms.Framework.Settings.CustomPropertyProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `CustomPropertyProperty` are:

  - Id

UserCustomPropertyData

Namespace

```
Ektron.Cms
```

Properties

- `ContentLanguage`

  ```
  public int ContentLanguage { set; get; }
  ```

- `ID`

  ```
  public long ID { set; get; }
  ```

- `Name`

  ```
  public string Name { set; get; }
  ```

- `ObjectType`

  ```
  public EkEnumeration.CMSObjectTypes ObjectType { get; }
  ```

- PropertyDisplayValueType

```
public EkEnumeration.ObjectPropertyDisplayTypes
  PropertyDisplayValueType { set; get; }
```

- PropertyValidationJSCall

```
public string PropertyValidationJSCall { set; get; }
```

- PropertyValidationMaxVal

```
public object PropertyValidationMaxVal { set; get; }
```

- PropertyValidationMessage

```
public string PropertyValidationMessage { set; get; }
```

- PropertyValidationMinVal

```
public object PropertyValidationMinVal { set; get; }
```

- PropertyValidationSelectList

```
public string PropertyValidationSelectList { set; get; }
```

- PropertyValidationType

```
public int PropertyValidationType { set; get; }
```

- PropertyValueType

```
public EkEnumeration.ObjectPropertyValueTypes
  PropertyValueType { set; get; }
```

- Required

```
public bool Required { set; get; }
```

DXH

8.60 and higher

The DXH manager category is a sub-category of Settings and manages connections between the Digital Experience Hub (DXH) and other applications. It has 1 class.

- *DXHUserConnectionManager* below. Manages connections between the Digital Experience Hub (DXH) and other applications.

DXHUserConnectionManager

The DXHUserConnectionManager class manages connections between the Digital Experience Hub (DXH) and other applications.

8.60 and higher

Namespace

```
Ektron.Cms.Framework.Settings.DxH.DxHUserConnectionManager
```

Constructors

- `DxHUserConnectionManager()`
- `DxHUserConnectionManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `DxHUserConnectionDataService`. Gets instance to DxHUserConnectionManager.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on the facing page
- *GetItem* on page 1473
- *GetList* on page 1474
- *Update* on page 1477

Add

```
Add(Ektron.Cms.Settings.UrlAliasing.DataObjects.RedirectData)
```

Adds a DxHUserConnectionData object with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * DXH Connection Code

Parameters

- `data`. DxHUserConnectionData object.

Returns

Returns added DxHUserConnectionData object.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            DxHConnectionManager dxhconnectionManager = new DxHConnectionManager();

            if (uxAdapterName.Text.Trim() == "" || uxConnectionName.Text.Trim() == "")
            {
                MessageUtilities.UpdateMessage(uxMessage, "Enter the required fields",
Message.DisplayModes.Error);
            }
            else
            {

                DxHConnectionData dxhConnectionData = new DxHConnectionData()
                {
                    AdapterName = uxAdapterName.Text,
                    ConnectionName = uxConnectionName.Text,
                    EndPoint = uxEndPoint.Text,

                };
                dxhconnectionManager.Add(dxhConnectionData);
                MessageUtilities.UpdateMessage(uxMessage, "DxConnection property added
with Id " + dxhConnectionData.Id, Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a DxHUserConnectionData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The ID of the DxHUserConnectionData to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxDxHUserConnectionIdLabel"
AssociatedControlID="uxDxHUserConnectionId"
            CssClass="span-4 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxDxHUserConnectionId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Settings.DxH;
using Ektron.Cms.Framework.Settings.DxH;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long ID = long.Parse(uxId.Text);
        DxHConnectionManager dxhconnectionManager = new DxHConnectionManager();
        DxHConnectionData dxhConnectionData = dxhconnectionManager.GetItem(ID);
        bool result = Information.IsNumeric(uxId.Text);
        if (dxhConnectionData != null)
        {
            dxhconnectionManager.Delete(ID);
            MessageUtilities.UpdateMessage(uxMessage, "DxHConnection deleted.",
Message.DisplayModes.Success);
        }
```

```
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "DxHConnection does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a RedirectData object by RedirectId.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Redirect ID

Parameters

- `Redirectid`. The ID of the task category to retrieve.

Returns

Returns a RedirectData object.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            DxHConnectionManager dxhconnectionManager = new DxHConnectionManager();
            long ID = long.Parse(uxID.Text);
            DxHConnectionData dxhConnectionData = dxhconnectionManager.GetItem(ID);

            if (dxhConnectionData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for user
DxHConnection Property with ID " + dxhConnectionData.Id.ToString() + " are below",
Message.DisplayModes.Success);

                uxAdapterName.Text = "AdapterName : " + dxhConnectionData.AdapterName;
```

```
                uxConnectionName.Text = "ConnectionName : " +
dxhConnectionData.ConnectionName;
                uxEndPoint.Text = "EndPoint : " + dxhConnectionData.EndPoint;
                uxCreateDate.Text = "CreateDate : " + dxhConnectionData.CreateDate;
                uxModifiedDate.Text = "uxModifiedDate : " +
dxhConnectionData.ModifiedDate;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "DxHConnection Property does
not exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Common.Criteria)
```

Retrieves a list of DxHUserConnectionData objects based upon the supplied criteria.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * DXH User Connection Property
- * Object Value

Parameters

- `criteria`. Criteria used to retrieve the DxHUserConnectionData.

Returns

Returns a list of DxHUserConnectionData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxDxHUserConnectionPropertyLabel"
AssociatedControlID="uxDxHUserConnectionProperty"
```

```
            CssClass="span-6 last" runat="server" Text="* DxHUserConnectionProperty:" />
        <asp:DropDownList ID="uxDxHUserConnectionProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>ConnectorName</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last"
            runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxDxHUserConnectionDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Visitor Id
                    </th>
                    <th>
                        Connector Name
                    </th>
                    <th>
                        Connector Object Name
                    </th>
                    <th>
                        Connector Id
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
```

```
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("VisitorId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ConnectorName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ConnectorObjectName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ConnectorId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Settings.DxH;
using Ektron.Cms.Framework.Settings.DxH;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
         object Objectvalue;
        DxHConnectionManager dxhconnectionManager = new DxHConnectionManager();
        DxHConnectionCriteria dxhConnectionCriteria = new DxHConnectionCriteria();
        string Property = uxDxHConnectionProperty.SelectedItem.Text;

        Objectvalue = uxObjectValue.Text;

        if (Property == "Id")
        {
            dxhConnectionCriteria.AddFilter(DxHConnectionProperty.Id,
CriteriaFilterOperator.EqualTo, Convert.ToInt64(Objectvalue));
        }
        else if (Property == "AdapterName")
        {
            dxhConnectionCriteria.AddFilter(DxHConnectionProperty.AdapterName,
CriteriaFilterOperator.EqualTo, Objectvalue);
        }
        else if (Property == "ConnectionName")
        {
            dxhConnectionCriteria.AddFilter(DxHConnectionProperty.ConnectionName,
```

```
CriteriaFilterOperator.EqualTo, Objectvalue);
        }
        else if (Property == "EndPoint")
        {
            dxhConnectionCriteria.AddFilter(DxHConnectionProperty.EndPoint,
CriteriaFilterOperator.EqualTo, Objectvalue);
        }


        //Get Activity List for given Filter
        List<DxHConnectionData> dxhConnectionlist = dxhconnectionManager.GetList
(dxhConnectionCriteria);

        uxDxHConnectionlist.Visible = true;
        uxDxHConnectionlist.DataSource = dxhConnectionlist;
        uxDxHConnectionlist.DataBind();

        uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Settings.UrlAliasing.DataObjects.RedirectData)
```

Updates the supplied DxHUserConnectionData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Redirect ID
- Redirect Code
- Original URL
- New URL
- Active

Parameters

- `data`. The DxHUserConnectionData object to update.

Returns

Updated the DxHUserConnectionData object.

Remarks

Validate the redirect item with `GetItem()` before you update the properties. Then, call `Update` with your modified DxHUserConnectionData object.

NOTE: You cannot change all properties after the initial Add event, such as the DxHUserConnectionData.Type.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            DxHConnectionManager dxhconnectionManager = new DxHConnectionManager();
            long Id = long.Parse(uxId.Text.Trim());

            DxHConnectionData dxhConnectionData = dxhconnectionManager.GetItem(Id);
            if (dxhConnectionData != null)
            {
                if (uxAdapterName.Text.Trim() == "" || uxConnectionName.Text.Trim() ==
"")
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Enter the required
fields", Message.DisplayModes.Error);
                }
                else
                {
                    dxhConnectionData.AdapterName = uxAdapterName.Text;
                    dxhConnectionData.ConnectionName = uxConnectionName.Text;
                    dxhConnectionData.EndPoint = uxEndPoint.Text;

                    dxhconnectionManager.Update(dxhConnectionData);
                    MessageUtilities.UpdateMessage(uxMessage, "DxHConnectionManager
updated", Message.DisplayModes.Success);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "DxHConnectionManager does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

DxHUserConnectionData

Namespace

```
Ektron.Cms.Settings.DxH
```

Properties

- `ConnectionName`. Gets or Sets the ConnectionName for DxHUserConnectionData object.

  ```
  public string ConnectionName { set; get; }
  ```

- `ConnectorName`. Gets or Sets the Connector Name for DxHUserConnectionData object.

  ```
  public string ConnectorName { set; get; }
  ```

- `ConnectorObjectName`. Gets or Sets the Connector Object Name for DxHUserConnectionData object.

  ```
  public string ConnectorObjectName { set; get; }
  ```

- `ExternalUserKey`. Gets or Sets the Id for the user in the external system.

  ```
  public string ExternalUserKey { set; get; }
  ```

- `Id`. Gets or Sets the Id for DxHUserConnectionData object.

  ```
  public long Id { set; get; }
  ```

- `VisitorId`Gets or Sets the VisitorId for DxHUserConnectionData object.

  ```
  public string VisitorId { set; get; }
  ```

Notifications

8.50 and higher

The Notifications manager category is a sub-category of Settings and manages notifications with the following classes:

- *NotificationAgentSettingManager* on the next page. Manages notification agents.
- *NotificationPreferenceManager* on page 1498. Manages preferences for notification agents.
- *NotificationPublishPreferenceManager* on page 1523. Manages preferences for notification agents that can be published.
- *UserNotificationSettingManager* on page 1535. Manages user-specified notifications.

NotificationAgentSettingManager

8.50 and higher

The NotificationAgentSettingManager class manages notification agents.

A notification agent is a plug-able component that sends notification messages. Each Agent is responsible for sending notifications to users when CMS activity occurs. After a Notification Agent is created and registered, users can set up notification preferences that use it.

For information about notifications, see Sending Notifications to a Community in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Settings.Notifications.NotificationAgentSettingManager
```

Constructors

- `NotificationAgentSettingManager()`
- `NotificationAgentSettingManager(ApiAccessMode)`

Properties

- `AgentSettingService`. Notification Agent Service
- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *GetRegisteredAgentList* on page 1492
- *Update* on page 1492

Add

```
Add(Ektron.Cms.Notifications.NotificationAgentData)
```

Adds a notification agent with details from the supplied NotificationAgentData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Name
- * Description
- Enable Agent
- * Type Name

Parameters

- `agentData`. Agent settings to save.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-4
last"
            runat="server" Text="*Name :" />
        <ektronUI:TextField ID="uxName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last"
            runat="server" Text="*Description :" />
        <ektronUI:TextField ID="uxDescription" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnableLabel" AssociatedControlID="uxIsEnable"
CssClass="span-4 last"
            runat="server" Text="Enable Agent:" />
        <asp:DropDownList ID="uxIsEnable" runat="server" CssClass="span-4 last">
```

```
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTypeNameLabel" AssociatedControlID="uxTypeName"
CssClass="span-4 last"
            runat="server" Text="*Type Name :" />
        <asp:DropDownList ID="uxTypeName" runat="server" CssClass="span-4 last">
            <asp:ListItem
Value="Ektron.Cms.Notifications.Providers.EktronEmailAgent">EktronEmail</asp:ListItem>
            <asp:ListItem
Value="Ektron.Cms.Notifications.Providers.ActivityStreamAgent">ActivityStream</asp:ListI
tem>
            <asp:ListItem
Value="Ektron.Cms.Notifications.Providers.SMSAgent">SMSAgent</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationAgentSettingManager notificationAgentSettingManager = new
NotificationAgentSettingManager();
            bool status = false;
            if (uxIsEnable.SelectedValue.ToString() == "True")
                status = true;

            NotificationAgentData notificationAgentData = new NotificationAgentData()
            {
                Name = uxName.Text,
                Description = uxDescription.Text,
                IsEnabled = status,
```

```
                TypeName = uxTypeName.SelectedValue,
            };
            notificationAgentSettingManager.Add(notificationAgentData);

            MessageUtilities.UpdateMessage(uxMessage,  "Notification Agent Saved with Id
" + notificationAgentData.Id, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a notification agent from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Notification Agent ID

Parameters

- `id`. ID of agent to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNotificationAgentIDLabel"
AssociatedControlID="uxNotificationAgentID" CssClass="span-4 last"
            runat="server" Text="*NotificationAgent ID :" />
        <ektronUI:TextField ID="uxNotificationAgentID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
```

```
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationAgentSettingManager notificationAgentSettingManager = new
NotificationAgentSettingManager();

            long NotificationAgentID = long.Parse(uxNotificationAgentID.Text);
            //Check Whether NotificationAgentID valid or Not
            NotificationAgentData notificationAgentData =
notificationAgentSettingManager.GetItem(NotificationAgentID);
            if (notificationAgentData != null)
            {  //Delete NotificationAgentID
                notificationAgentSettingManager.Delete(NotificationAgentID);
                MessageUtilities.UpdateMessage(uxMessage, "Notification Agent  with Id "
+ NotificationAgentID + " has been deleted from the CMS", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid
NotificationAgent ID and try again", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetAgent

```
GetAgent(System.Int64)
```

Retrieves the Notification Agent based on the Notification Agent ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Notification Agent ID

Parameters

- `id`. ID of agent to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNotificationAgentIDLabel"
AssociatedControlID="uxNotificationAgentID" CssClass="span-4 last"
            runat="server" Text="*NotificationAgent ID :" />
        <ektronUI:TextField ID="uxNotificationAgentID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetAgent"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAgentID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxNotificationAgentName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxNotificationAgentDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsEnabled" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationAgentSettingManager notificationAgentSettingManager = new
NotificationAgentSettingManager();

            long NotificationAgentID = long.Parse(uxNotificationAgentID.Text);
            //Check  Whether NotificationAgentID valid or Not
            NotificationAgentData notificationAgentData =
notificationAgentSettingManager.GetItem(NotificationAgentID);
            if (notificationAgentData != null)
            {
                NotificationAgent notificationAgent =
notificationAgentSettingManager.GetAgent(NotificationAgentID);
                if (notificationAgent != null)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Notification Agent
Details for Id " + notificationAgentData.Id + " are below",
Message.DisplayModes.Success);

                    uxAgentID.Text = "Notification Agent ID: " +
notificationAgentData.Id;
                    uxNotificationAgentName.Text = "Actual Agent Name: " +
notificationAgent.Name;
                    uxNotificationAgentDescription.Text = "Actual Agent Description: " +
notificationAgent.Description;
                    uxIsEnabled.Text = "IsValidationRequired: " +
notificationAgent.IsValidationRequired;
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage,"There is No Actual
Notification Agent defined in the system for Id " + NotificationAgentID + ".",
Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid
NotificationAgent ID and try again", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
```

```
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.NotificationAgentData)
```

Retrieves the properties of a specific NotificationAgentData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Notification Agent ID

Parameters

- `id`. ID of agent to retrieve settings for.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNotificationAgentIDLabel"
AssociatedControlID="uxNotificationAgentID" CssClass="span-4 last"
            runat="server" Text="*NotificationAgent ID :" />
        <ektronUI:TextField ID="uxNotificationAgentID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAgentID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxNotificationAgentName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxNotificationAgentDescription" runat="server"></asp:Literal>
```

```
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxNotificationAgentType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsEnabled" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationAgentSettingManager notificationAgentSettingManager = new
NotificationAgentSettingManager();

            long NotificationAgentID = long.Parse(uxNotificationAgentID.Text);
            //Check  Whether NotificationAgentID valid or Not
            NotificationAgentData notificationAgentData =
notificationAgentSettingManager.GetItem(NotificationAgentID);
            if (notificationAgentData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Notification Agent  Details
for Id " + notificationAgentData.Id + " are below", Message.DisplayModes.Success);

                uxAgentID.Text = "Notification Agent ID: " + notificationAgentData.Id;
                uxNotificationAgentName.Text = "Notification Agent Name: " +
notificationAgentData.Name;
                uxNotificationAgentDescription.Text = "Notification Agent Description: "
+ notificationAgentData.Description;
                uxNotificationAgentType.Text = "Notification Agent TypeName: " +
notificationAgentData.TypeName;
                uxIsEnabled.Text = "Notification Agent IsEnable: " +
notificationAgentData.IsEnabled;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid
NotificationAgent ID and try again", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(NotificationAgentCriteria)
```

Retrieves lists of objects through the GetList(NotificationAgentCriteria) method.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Notification Agent Property
- * Object Value

Parameters

- `criteria`. Criteria by which to retrieve agent settings.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNotificationAgentPropertyLabel"
AssociatedControlID="uxNotificationAgentProperty"
            CssClass="span-5 last" runat="server" Text="NotificationAgent Property :" />
        <asp:DropDownList ID="uxNotificationAgentProperty" runat="server">
            <asp:ListItem>TypeName
(EktronEmailAgent,ActivityStreamAgent,SMSAgent)</asp:ListItem>
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last"
            runat="server" Text="ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
```

```
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxNotificationAgentListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Name
                    </th>
                    <th>
                        Description
                    </th>
                    <th>
                        TypeName
                    </th>
                    <th>
                        IsEnabled
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Description")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TypeName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsEnabled")%>
            </td>
        </tr>
    </ItemTemplate>
```

```
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationAgentSettingManager notificationAgentSettingManager = new
NotificationAgentSettingManager();
            object Objectvalue;
            string Property = uxNotificationAgentProperty.SelectedItem.Text;

            NotificationAgentCriteria criteria = new NotificationAgentCriteria
(NotificationAgentProperty.Id, EkEnumeration.OrderByDirection.Ascending);
            if (Property == "Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(NotificationAgentProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (Property == "Name")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(NotificationAgentProperty.Name,
CriteriaFilterOperator.Contains, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(NotificationAgentProperty.TypeName,
CriteriaFilterOperator.Contains, Objectvalue);
            }
            List<NotificationAgentData> NotificationAgentList =
notificationAgentSettingManager.GetList(criteria);

            uxNotificationAgentListView.Visible = true;
            uxNotificationAgentListView.DataSource = NotificationAgentList;
            uxNotificationAgentListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

```
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetRegisteredAgentList

```
GetRegisteredAgentList
```

Retrieves a list of the notification agents registered in `web.config`.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationAgentSettingManager notificationAgentSettingManager = new
NotificationAgentSettingManager();
            if (uxAgentList.Checked)
            {
                List<NotificationAgent> NotificationAgentList =
notificationAgentSettingManager.GetRegisteredAgentList();
                uxNotificationAgentListView.Visible = true;
                uxNotificationAgentListView.DataSource = NotificationAgentList;
                uxNotificationAgentListView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Please Check the 'Get
Notification Agent List' CheckBox and Try again", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.NotificationAgentData)
```

Updates an existing NotificationAgentData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Notification Agent ID
- Name
- Description
- Enable Agent
- Type Name

Parameters

- `agentData`. Agent settings to save.

Returns

Returns the custom CmsData object updated.

Remarks

Validate the NotificationAgent item with `GetItem()` before you update the properties. Then, call `Update` with your modified NotificationAgentData object.

NOTE: You cannot change all properties after the initial Add event, such as the NotificationAgentData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNotificationAgentIDLabel"
AssociatedControlID="uxNotificationAgentID" CssClass="span-4 last"
            runat="server" Text="*NotificationAgent ID :" />
        <ektronUI:TextField ID="uxNotificationAgentID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-4
last"
            runat="server" Text="Name :" />
        <ektronUI:TextField ID="uxName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-4 last"
```

```
            runat="server" Text="Description :" />
        <ektronUI:TextField ID="uxDescription" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnableLabel" AssociatedControlID="uxIsEnable"
CssClass="span-4 last"
            runat="server" Text="Enable Agent:" />
        <asp:DropDownList ID="uxIsEnable" runat="server" CssClass="span-4 last">
            <asp:ListItem>SelectOne</asp:ListItem>
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTypeNameLabel" AssociatedControlID="uxTypeName"
CssClass="span-4 last"
            runat="server" Text="*Type Name :" />
        <asp:DropDownList ID="uxTypeName" runat="server" CssClass="span-4 last">
            <asp:ListItem Value="">SelectOne</asp:ListItem>
            <asp:ListItem
Value="Ektron.Cms.Notifications.Providers.EktronEmailAgent">EktronEmail</asp:ListItem>
            <asp:ListItem
Value="Ektron.Cms.Notifications.Providers.ActivityStreamAgent">ActivityStream</asp:ListI
tem>
            <asp:ListItem
Value="Ektron.Cms.Notifications.Providers.SMSAgent">SMSAgent</asp:ListItem>
        </asp:DropDownList>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
```

```
        {
            NotificationAgentSettingManager notificationAgentSettingManager = new
NotificationAgentSettingManager();
            bool status = false;

            long NotificationAgentID = long.Parse(uxNotificationAgentID.Text);
            //Check Whether NotificationAgentID valid or Not
            NotificationAgentData notificationAgentData =
notificationAgentSettingManager.GetItem(NotificationAgentID);
            if (notificationAgentData != null)
            {
                if (uxIsEnable.SelectedValue.ToString() != "SelectOne")
                {
                    if (uxIsEnable.SelectedValue.ToString() == "True")
                        status = true;
                }
                else
                {
                    status = notificationAgentData.IsEnabled;
                }
                notificationAgentData.Name = uxName.Text != string.Empty ? uxName.Text :
notificationAgentData.Name;
                notificationAgentData.Description = uxDescription.Text != string.Empty ?
uxDescription.Text : notificationAgentData.Description;
                notificationAgentData.TypeName = uxTypeName.SelectedValue.ToString() !=
string.Empty ? uxTypeName.SelectedValue.ToString() : notificationAgentData.TypeName;
                notificationAgentData.IsEnabled = status;
                //Update notificationAgentData with given Details
                notificationAgentSettingManager.Update(notificationAgentData);

                MessageUtilities.UpdateMessage(uxMessage, "Notification Agent  with Id "
+ notificationAgentData.Id + " has been Updated", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid
NotificationAgent ID and try again", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

NotificationAgentCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Notifications
```

Constructors

- `NotificationAgentCriteria()`

  ```
  public NotificationAgentCriteria()
  ```

- `NotificationAgentCriteria (Ektron.Cms.Notifications.NotificationAgentProperty, EkEnumeration.OrderByDirection)`

  ```
  public NotificationAgentCriteria(Ektron.Cms.Notifications.NotificationAgentProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `NotificationAgentProperty` are:

  - Description
  - Id
  - IsEnabled
  - Name
  - TypeName

NotificationAgentData

Namespace

```
Ektron.Cms.Notificiations
```

Properties

- `NotificationAgentData()`

  ```
  public NotificationAgentData()
  ```

- `NotificationAgentData(long, string, string, string, bool)`

  ```
  public NotificationAgentData(long id, string name,
    string description, string typeName, bool isEnabled)
  ```

- `Validate()`. Validates that the data object is valid for saving. Any errors are returns as ValidationResults.

  ```
  public ValidationResults Validate()
  ```

- `Description`. Gets or sets the description of the agent.

  ```
  public string Description { set; get; }
  ```

- `Id`. Get or sets the ID of the notification agent.

  ```
  public long Id { set; get; }
  ```

- `IsEnabled`. Gets or sets the IsEnabled flag on the agent.

  ```
  public bool IsEnabled { set; get; }
  ```

- `Name`. Gets or sets the Name of the agent.

  ```
  public string Name { set; get; }
  ```

- `TypeName`. Gets or sets the fully qualified name of the agent type.

```
public string TypeName { set; get; }
```

NotificationPreferenceManager

8.50 and higher

The NotificationPreferenceManager class manages preferences for notification agents.

A notification agent is a plug-able component that sends notification messages. Each Agent is responsible for sending notifications to users when CMS activity occurs. Once a Notification Agent is created and registered, users can set up notification preferences that use it.

For information about notifications, see Sending Notifications to a Community in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Settings.Notifications.NotificationPreferenceManager
```

Constructors

- `NotificationPreferenceManager()`
- `NotificationPreferenceManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `NotificationPreferenceService`. Notification preference service
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *GetUserNotificationListForActivity* on page 1510
- *SaveUserPreferences* on page 1513
- *Update* on page 1517

Add

```
Add(Ektron.Cms.Notifications.NotificationPreferenceData)
```

Adds a notification agent with details from the supplied NotificationPreferenceData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Agent ID
- * Activity Type ID
- Object ID

Parameters

- `preferenceData`. Preference data to save.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="*UserId :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAgentIdLabel" AssociatedControlID="uxAgentId"
CssClass="span-4 last"
            runat="server" Text="*AgentId :" />
        <ektronUI:TextField ID="uxAgentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="2" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityTypeIdLabel"
```

```
AssociatedControlID="uxActivityTypeId"
            CssClass="span-4 last" runat="server" Text="*ActivityTypeId:" />
        <ektronUI:TextField ID="uxActivityTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="20" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectIdLabel" AssociatedControlID="uxObjectId"
CssClass="span-4 last"
            runat="server" Text="Object Id:" />
        <ektronUI:TextField ID="uxObjectId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="30" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPreferenceManager notificationPreferenceManager = new
NotificationPreferenceManager();
            //Set the Preference with given Details
            NotificationPreferenceData notificationPreferenceData = new
NotificationPreferenceData()
            {
                ActivityTypeId = long.Parse(uxActivityTypeId.Text),//ActivityTypeId=20
is for 'Add Site Content'.
                AgentId = long.Parse(uxAgentId.Text),//AgentId =2 is for 'Activity
Stream.'
                UserId = long.Parse(uxUserId.Text),
                ObjectId = long.Parse(uxObjectId.Text),
            };
            notificationPreferenceManager.Add(notificationPreferenceData);
```

```
            MessageUtilities.UpdateMessage(uxMessage,  "Notification Preference Saved
with Id " + notificationPreferenceData.Id, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a notification preference from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Notification Preference ID

Parameters

- `id`. ID of notification preference to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxnotificationPreferenceIdLabel"
AssociatedControlID="uxnotificationPreferenceId"
            CssClass="span-5 last" runat="server" Text="*NotificationPreference Id :" />
        <ektronUI:TextField ID="uxnotificationPreferenceId" CssClass="span-6"
runat="server"
            ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>

    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPreferenceManager notificationPreferenceManager = new
NotificationPreferenceManager();
            long notificationPreferenceId = long.Parse(uxnotificationPreferenceId.Text);

            //Check Whether Entered notificationPreference ID is Valid or Not
            NotificationPreferenceData notificationPreferenceData =
notificationPreferenceManager.GetItem(notificationPreferenceId);
            if (notificationPreferenceData != null)
            {
                //Delete notificationPreference ID from the CMS.
                notificationPreferenceManager.Delete(notificationPreferenceId);

                MessageUtilities.UpdateMessage(uxMessage, "Notification Preference with
Id " + notificationPreferenceId + " has been Deleted from the CMS",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid
Notification Preference ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetDefaultPreferenceList

```
GetDefaultPreferenceList
(Ektron.Cms.Notifications.NotificationPreferenceCriteria)
```

Retrieves a list of default notification preferences based on the supplied criteria.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Notification Preference Property
- * Object Value

Parameters

- `criteria`. NotificationPreferenceCriteria used to determine which default preferences to return.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNotificationPreferencePropertyLabel"
AssociatedControlID="uxNotificationPreferenceProperty"
            CssClass="span-5 last" runat="server" Text="NotificationPreference Property
:" />
        <asp:DropDownList ID="uxNotificationPreferenceProperty" runat="server">
            <asp:ListItem>UserId</asp:ListItem>
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>ActivityTypeId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last"
            runat="server" Text="ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxNotificationPreferenceListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
```

```
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        UserId
                    </th>
                    <th>
                        ActivityTypeId
                    </th>
                    <th>
                        ObjectId
                    </th>
                    <th>
                        AgentId
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ActivityTypeId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("AgentId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
```

```
using Ektron.Cms.Notifications;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPreferenceManager notificationPreferenceManager = new
NotificationPreferenceManager();

            object Objectvalue;
            string Property = uxNotificationPreferenceProperty.SelectedItem.Text;
            Objectvalue = Convert.ToInt64(uxObjectValue.Text);

            NotificationPreferenceCriteria criteria = new NotificationPreferenceCriteria
(NotificationPreferenceProperty.Id, EkEnumeration.OrderByDirection.Ascending);
            if (Property == "Id")
            {
                criteria.AddFilter(NotificationPreferenceProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (Property == "UserId")
            {
                criteria.AddFilter(NotificationPreferenceProperty.UserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                criteria.AddFilter(NotificationPreferenceProperty.ActivityTypeId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            List<NotificationPreferenceData> NotificationPreferenceList =
notificationPreferenceManager.GetDefaultPreferenceList(criteria);

            uxNotificationPreferenceListView.Visible = true;
            uxNotificationPreferenceListView.DataSource = NotificationPreferenceList;
            uxNotificationPreferenceListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.NotificationPreferenceData)
```

Retrieves the properties of a specific NotificationPreferenceData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Notification Preference ID

Parameters

- `id`. ID of Notification Preference to return.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxnotificationPreferenceIdLabel"
AssociatedControlID="uxnotificationPreferenceId"
            CssClass="span-5 last" runat="server" Text="*NotificationPreference Id :" />
        <ektronUI:TextField ID="uxnotificationPreferenceId" CssClass="span-6"
runat="server"
            ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>

    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxActivityTypeId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAgentId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxObjectId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPreferenceManager notificationPreferenceManager = new
NotificationPreferenceManager();
            long notificationPreferenceId = long.Parse(uxnotificationPreferenceId.Text);

            //Check Whether Entered notificationPreference ID is Valid or Not
            NotificationPreferenceData notificationPreferenceData =
notificationPreferenceManager.GetItem(notificationPreferenceId);
            if (notificationPreferenceData != null)
            {
                //Display the notificationPreference Details
                MessageUtilities.UpdateMessage(uxMessage, "Notification Preference with
Id " + notificationPreferenceId + " details are below:", Message.DisplayModes.Success);

                uxUserID.Text = "UserId : " + notificationPreferenceData.UserId;
                uxActivityTypeId.Text = "Activity TypeId : " +
notificationPreferenceData.ActivityTypeId;
                uxAgentId.Text = "Agent ID : " + notificationPreferenceData.AgentId;
                uxObjectId.Text = "Object ID : " + notificationPreferenceData.ObjectId;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Please Enter Valid
Notification Preference ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(NotificationPreferenceCriteria)
```

Retrieves lists of objects through the GetList(NotificationPreferenceCriteria) method.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Notification Preference Property
- * Object Value

Parameters

- `criteria`. Criteria used to determine which preferences to return.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNotificationPreferencePropertyLabel"
AssociatedControlID="uxNotificationPreferenceProperty"
            CssClass="span-5 last" runat="server" Text="NotificationPreference Property
:" />
        <asp:DropDownList ID="uxNotificationPreferenceProperty" runat="server">
            <asp:ListItem>UserId</asp:ListItem>
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>ActivityTypeId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last"
            runat="server" Text="ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxNotificationPreferenceListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
```

```
                    <th>
                        UserId
                    </th>
                    <th>
                        ActivityTypeId
                    </th>
                    <th>
                        ObjectId
                    </th>
                    <th>
                        AgentId
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ActivityTypeId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ObjectId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("AgentId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPreferenceManager notificationPreferenceManager = new
NotificationPreferenceManager();

            object Objectvalue;
            string Property = uxNotificationPreferenceProperty.SelectedItem.Text;
            Objectvalue = Convert.ToInt64(uxObjectValue.Text);

            NotificationPreferenceCriteria criteria = new NotificationPreferenceCriteria
(NotificationPreferenceProperty.Id, EkEnumeration.OrderByDirection.Ascending);
            if (Property == "Id")
            {
                criteria.AddFilter(NotificationPreferenceProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (Property == "UserId")
            {
                criteria.AddFilter(NotificationPreferenceProperty.UserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                criteria.AddFilter(NotificationPreferenceProperty.ActivityTypeId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            List<NotificationPreferenceData> NotificationPreferenceList =
notificationPreferenceManager.GetList(criteria);

            uxNotificationPreferenceListView.Visible = true;
            uxNotificationPreferenceListView.DataSource = NotificationPreferenceList;
            uxNotificationPreferenceListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetUserNotificationListForActivity

```
GetUserNotificationListForActivity(Ektron.Cms.Activity.ActivityData,
Ektron.Cms.PagingInfo)
```

Retrieves a list of users and their notification settings for a given activity occurrence.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Activity ID

Parameters

- `activity`. Activity definition to retrieve notifications for.
- `paging`. Paging information for the retrieval.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityIdLabel" AssociatedControlID="uxActivityId"
CssClass="span-5 last"
            runat="server" Text="*Activity Id :" />
        <ektronUI:TextField ID="uxActivityId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxuserNotificationListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        User
                    </th>
                    <th>
                        AgentName
                    </th>
                    <th>
                        AgentSettings
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
```

```
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("User")%>
            </td>
            <td class="devsite-method">
                <%# Eval("AgentName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("AgentSettings")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPreferenceManager notificationPreferenceManager = new
NotificationPreferenceManager();
            ActivityManager activityManager = new ActivityManager();
            long ActivityId = long.Parse(uxActivityId.Text);

            //Check Whether Entered Activity ID is Valid or Not
            ActivityData activityData = activityManager.GetItem(ActivityId);
            if (activityData != null)
            {
                PagingInfo pagingInfo = new PagingInfo();
                List<UserNotificationData> userNotificationList =
notificationPreferenceManager.GetUserNotificationListForActivity(activityData,
pagingInfo);

                //Display the userNotification List
                uxuserNotificationListView.Visible = true;
                uxuserNotificationListView.DataSource = userNotificationList;
                uxuserNotificationListView.DataBind();
```

```
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Activity
ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

SaveUserPreferences

```
SaveUserPreferences(System.Collections.Generic.List)
```

Saves a set of notification preferences for a user. Preferences marked as added are added and those marked as deleted are removed.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * List of Users
- Notify me about these colleague activities (SMS, activity stream, and Email checklists)

Parameters

- `preferenceList`. Preference data to save.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="Label2" AssociatedControlID="aListOfUsers" CssClass="span-5
last"
            runat="server" Text="*List of Users:" />
        <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="true"
OnSelectedIndexChanged="aListOfUsers_SelectedIndexChanged" />
    </li>
    <li>
        <div id="Div1" class="EkMembershipActivityTab">
            <div class="dvCollGrid" id="Div2">
                <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
CssClass="ektronGrid"
                GridLines="None">
```

```
                <HeaderStyle CssClass="title-header" />
                    <Columns>
                        <asp:TemplateField HeaderText="">
                            <ItemTemplate>
                                <asp:Literal ID="DisplayName" runat="server" />
                            </ItemTemplate>
                        </asp:TemplateField>
                        <asp:TemplateField HeaderText="SMS">
                            <ItemTemplate>
                                <asp:CheckBox ID="ChkBxSMS" runat="server" />
                            </ItemTemplate>
                        </asp:TemplateField>
                        <asp:TemplateField HeaderText="Activity Stream">
                            <ItemTemplate>
                                <asp:CheckBox ID="ChkBxActivityStream" runat="server" />
                            </ItemTemplate>
                        </asp:TemplateField>
                        <asp:TemplateField HeaderText="Email">
                            <ItemTemplate>
                                <asp:CheckBox ID="ChkBxEmail" runat="server" />
                            </ItemTemplate>
                        </asp:TemplateField>
                    </Columns>
                </asp:GridView>
            </div>
        </div>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="Button1" runat="server" OnClick="uxSubmit_Click"
Text="Save"></ektronUI:Button>
        <ektronUI:Label ID="Label2" CssClass="span-4" runat="server" Text="* - Save
Preferences" />
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Activity;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Notifications;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
{
    int userId = GetUserId();
```

```
    List<NotificationPreferenceData> updatedPreferenceList = new
List<NotificationPreferenceData>();
    IList<NotificationPreferenceData> preferenceList = LoadPreferenceList(userId);
    List<ActivityTypeData> activityTypeData = GetActivityList();
    const int emailAgentId = 1;
    const int newsFeedAgentId = 2;
    const int smsAgentId = 3;

    foreach (GridViewRow row in CollGrid.Rows)

      Literal name = row.FindControl("DisplayName") as Literal;
      CheckBox chkBxSMS = row.FindControl("ChkBxSMS") as CheckBox;
      CheckBox chkBxActivityStream = row.FindControl("ChkBxActivityStream") as CheckBox;
      CheckBox chkBxEmail = row.FindControl("ChkBxEmail") as CheckBox;

      if (name == null || chkBxSMS == null || chkBxEmail == null || chkBxActivityStream
== null) return;

      ActivityTypeData foundActivityType = activityTypeData.SingleOrDefault(x => x.Name
== name.Text);

      if (foundActivityType == null) return;

      NotificationPreferenceData emailNotification = preferenceList.SingleOrDefault(x =>
x.ActivityTypeId == foundActivityType.Id && x.AgentId == emailAgentId);
      NotificationPreferenceData newsFeedNotification = preferenceList.SingleOrDefault(x
=> x.ActivityTypeId == foundActivityType.Id && x.AgentId == newsFeedAgentId);
      NotificationPreferenceData smsAgentNotification = preferenceList.SingleOrDefault(x
=> x.ActivityTypeId == foundActivityType.Id && x.AgentId == smsAgentId);

      #region Email

      if (chkBxEmail.Checked && emailNotification == null)
      {
        emailNotification = new NotificationPreferenceData
          {
            ActivityTypeId = foundActivityType.Id,
            AgentId = emailAgentId,
            DataState = EkEnumeration.DataState.Added,
            UserId = userId
          };
        updatedPreferenceList.Add(emailNotification);
       }
        else if (chkBxEmail.Checked && emailNotification.DataState ==
EkEnumeration.DataState.Deleted)
       {
        emailNotification = new NotificationPreferenceData
          {
            ActivityTypeId = foundActivityType.Id,
            AgentId = emailAgentId,
            DataState = EkEnumeration.DataState.Updated,
            UserId = userId
          };
        updatedPreferenceList.Add(emailNotification);
      }
      else if (!chkBxEmail.Checked && emailNotification != null)
```

```
      {
        emailNotification = new NotificationPreferenceData
          {
            ActivityTypeId = foundActivityType.Id,
            AgentId = emailAgentId,
            DataState = EkEnumeration.DataState.Deleted,
            UserId = userId
          };
        updatedPreferenceList.Add(emailNotification);
      }
      else if (!chkBxEmail.Checked && emailNotification == null)
      {
       // do nothing
      }
      #endregion
      #region ActivityStream

      if (chkBxActivityStream.Checked && newsFeedNotification == null)
      {
        newsFeedNotification = new NotificationPreferenceData
          {
            ActivityTypeId = foundActivityType.Id,
            AgentId = newsFeedAgentId,
            DataState = EkEnumeration.DataState.Added,
            UserId = userId
          };
        updatedPreferenceList.Add(newsFeedNotification);
      }
      else if (chkBxActivityStream.Checked && newsFeedNotification.DataState ==
EkEnumeration.DataState.Deleted)
      {
        newsFeedNotification = new NotificationPreferenceData
          {
            ActivityTypeId = foundActivityType.Id,
            AgentId = newsFeedAgentId,
            DataState = EkEnumeration.DataState.Updated,
            UserId = userId
          };
        updatedPreferenceList.Add(newsFeedNotification);
      }
      else if (!chkBxActivityStream.Checked && newsFeedNotification != null)
      {
        newsFeedNotification = new NotificationPreferenceData
          {
            ActivityTypeId = foundActivityType.Id,
            AgentId = newsFeedAgentId,
            DataState = EkEnumeration.DataState.Deleted,
            UserId = userId
          };
        updatedPreferenceList.Add(newsFeedNotification);
      }
      else if (!chkBxActivityStream.Checked && newsFeedNotification == null)
      {
        // do nothing
      }
      #endregion
```

```
      #region SMS
      if (chkBxSMS.Checked && smsAgentNotification == null)
      {
        smsAgentNotification = new NotificationPreferenceData
          {
            ActivityTypeId = foundActivityType.Id,
            AgentId = smsAgentId,
            DataState = EkEnumeration.DataState.Added,
            UserId = userId
          };
        updatedPreferenceList.Add(smsAgentNotification);
      }
      else if (chkBxSMS.Checked && smsAgentNotification.DataState ==
EkEnumeration.DataState.Deleted)
      {
        smsAgentNotification = new NotificationPreferenceData
          {
            ActivityTypeId = foundActivityType.Id,
            AgentId = smsAgentId,
            DataState = EkEnumeration.DataState.Updated,
            UserId = userId
          };
        updatedPreferenceList.Add(smsAgentNotification);
      }
      else if (!chkBxSMS.Checked && smsAgentNotification != null)
      {
        smsAgentNotification = new NotificationPreferenceData
          {
            ActivityTypeId = foundActivityType.Id,
            AgentId = smsAgentId,
            DataState = EkEnumeration.DataState.Deleted,
            UserId = userId
          };
        updatedPreferenceList.Add(smsAgentNotification);
      }
      else if (!chkBxSMS.Checked && smsAgentNotification == null)
      {
        // do nothing
      }
      #endregion
    }
    try
    {
      new NotificationPreferenceManager().SaveUserPreferences(updatedPreferenceList);
      MessageUtilities.UpdateMessage(uxMessage, "Saved User Preferences with given
Details", Message.DisplayModes.Success);
      uxPageMultiView.SetActiveView(uxViewMessage);
    }
    catch (Exception ex)
    {
      MessageUtilities.UpdateMessage(uxMessage, ex.Message, Message.DisplayModes.Error);
      uxPageMultiView.SetActiveView(uxViewMessage);
    }
  }
```

Update

```
Update(Ektron.Cms.NotificationAgentData)
```

Updates an existing notification preference item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Notification Preference ID
- User ID
- Agent ID
- Activity Type ID
- Object ID

Parameters

- `preferenceData`. Preference data to save.

Returns

Returns the custom CmsData object updated.

Remarks

Validate the notification preference item with `GetItem()` before you update the properties. Then, call `Update` with your modified NotificationPreferenceData object.

NOTE: You cannot change all properties after the initial Add event, such as the NotificationPreferenceData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxnotificationPreferenceIdLabel"
AssociatedControlID="uxnotificationPreferenceId"
            CssClass="span-5 last" runat="server" Text="*NotificationPreference Id :" />
        <ektronUI:TextField ID="uxnotificationPreferenceId" CssClass="span-6"
runat="server"
            ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-5 last"
            runat="server" Text="UserId :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxAgentIdLabel" AssociatedControlID="uxAgentId"
CssClass="span-5 last"
            runat="server" Text="AgentId :" />
        <ektronUI:TextField ID="uxAgentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityTypeIdLabel"
AssociatedControlID="uxActivityTypeId"
            CssClass="span-5 last" runat="server" Text="ActivityTypeId:" />
        <ektronUI:TextField ID="uxActivityTypeId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectIdLabel" AssociatedControlID="uxObjectId"
CssClass="span-5 last"
            runat="server" Text="Object Id:" />
        <ektronUI:TextField ID="uxObjectId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPreferenceManager notificationPreferenceManager = new
NotificationPreferenceManager();
            long notificationPreferenceId = long.Parse(uxnotificationPreferenceId.Text);

            //Check Whether Entered notificationPreference ID is Valid or Not
```

```
            NotificationPreferenceData notificationPreferenceData =
notificationPreferenceManager.GetItem(notificationPreferenceId);
            if (notificationPreferenceData != null)
            {
                notificationPreferenceData.UserId = long.Parse(uxUserId.Text) != 0 ?
long.Parse(uxUserId.Text) : notificationPreferenceData.UserId;
                notificationPreferenceData.AgentId = long.Parse(uxAgentId.Text) != 0 ?
long.Parse(uxAgentId.Text) : notificationPreferenceData.AgentId;
                notificationPreferenceData.ActivityTypeId = long.Parse
(uxActivityTypeId.Text) != 0 ? long.Parse(uxActivityTypeId.Text) :
notificationPreferenceData.ActivityTypeId;
                notificationPreferenceData.ObjectId = long.Parse(uxObjectId.Text) != 0 ?
long.Parse(uxObjectId.Text) : notificationPreferenceData.ObjectId;

                //Update notificationPreferenceData with given Details
                notificationPreferenceManager.Update(notificationPreferenceData);

                MessageUtilities.UpdateMessage(uxMessage, "Notification Preference with
Id " + notificationPreferenceData.Id + " has been Updated",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid
Notification Preference ID and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

NotificationPreferenceCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Notifications
```

Constructors

- `NotificationPreferenceCriteria()`

  ```
  public NotificationPreferenceCriteria()
  ```

- `NotificationPreferenceCriteria (Ektron.Cms.Notifications.NotificationPreferenceProperty, EkEnumeration.OrderByDirection)`

```
public NotificationPreferenceCriteria
(Ektron.Cms.Notifications.NotificationPreferenceProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `NotificationPreferenceProperty` are:

- ActionSource
- ActionSourceId
- ActivityActionType
- ActivityTypeId
- AgentId
- Id
- ObjectId
- ObjectType
- UserId

NotificationPreferenceData

Namespace

```
Ektron.Cms.Notifications
```

Properties

- `ActionSourceId`. Gets or sets the Object Id of the Notification Action Source for this notification preference. If `ActionSource = CommunityGroup`, `ActionSourceId` would be the community group ID you want to receive notifications from. By default it is 0, which means all.

  ```
  public long ActionSourceId { set; get; }
  ```

- `ActivityTypeId`. Gets or sets the activity type for this notification preference. Built in CMS activity are defined in `EkEnumeration.ActivityType`.

  ```
  public long ActivityTypeId { set; get; }
  ```

- `AgentId`. Gets or sets the Provider Id for this notification preference.

  ```
  public long AgentId { set; get; }
  ```

- `Clone()`

  ```
  public Ektron.Cms.Notifications.NotificationPreferenceData Clone()
  ```

- `DataState`. Gets or sets the preferences persistence State. Used to determine the state of a preference when saving batches.

  ```
  public EkEnumeration.DataState DataState { set; get; }
  ```

- `Id`. Gets or sets the Id for this notification preference.

  ```
  public long Id { set; get; }
  ```

- `NotificationPreferenceData()`

  ```
  public NotificationPreferenceData()
  ```

- `NotificationPreferenceData(long, long, long, long, long)`

```
public NotificationPreferenceData(long id, long userId,
  long activityTypeId, long objectId, long agentId,
  long actionSourceId)
```

- `ObjectId`. Gets or sets the Object Id for this notification preference. The Object Id is related to the Object Type.

```
public long ObjectId { set; get; }
```

- `UserId`. Gets or sets the User Id for this notification preference.

```
public long UserId { set; get; }
```

- `Validate()`. Validates the current data class is an appropriate state for saving. Any errors are returned.

```
public ValidationResults Validate()
```

NotificationPublishPreferenceManager

8.50 and higher

The NotificationPublishPreferenceManager class manages preferences for notification agents that can be published.

A notification agent is a plug-able component that sends notification messages. Each Agent is responsible for sending notifications to users when CMS activity occurs. Once a Notification Agent is created and registered, users can set up notification preferences that use it.

For information about notifications, see Sending Notifications to a Community in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Settings.Notifications.NotificationPublishPreferenceManager
```

Constructors

- `NotificationPublishPreferenceManager()`
- `NotificationPublishPreferenceManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Notifications.NotificationPreferenceData)
```

Sets the list of activity types for publishing for a given user. The method disables any Activity Types not listed.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Activity Type ID

Parameters

- `userId`. ID to add notification publishing.
- `ActivityTypeIdList`. List of Activity Type IDs to enable publishing for.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="*UserId :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="10" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityTypeIdsLabel"
AssociatedControlID="uxActivityTypeIds"
            CssClass="span-4 last" runat="server" Text="*ActivityTypeIds (Comma
Separated):" />
        <ektronUI:TextField ID="uxActivityTypeIds" CssClass="span-6"
ValidationGroup="RegisterValidationGroup"
            runat="server" Text="20,22" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPublishPreferenceManager notificationPublishPreferenceManager =
new NotificationPublishPreferenceManager();
            long userID = long.Parse(uxUserId.Text);

            string[] activityTypeIds = uxActivityTypeIds.Text.Split(',', ';');
            List<long> activityTypeIdList = new List<long>();

            UserManager Usermanager = new UserManager();
            //Get the User data for given UserID
            UserData Userdata = Usermanager.GetItem(userID);
            //Check Whether Entered User ID is valid or Not.
            if (Userdata != null)
            {
                ActivityTypeManager activityTypeManager = new ActivityTypeManager();
                for (int i = 0; i < activityTypeIds.Length; i++)
                {
                    long activityTypeID = long.Parse(activityTypeIds[i]);
                    //Check whether Entered activityTypeID valid or Not
                    ActivityTypeData activityTypeData = activityTypeManager.GetItem
(activityTypeID);
                    if (activityTypeData != null)
                    {
                        activityTypeIdList.Add(activityTypeID);
                        if (i == activityTypeIds.Length - 1)
                        {
                            notificationPublishPreferenceManager.Add(userID,
activityTypeIdList);
                            MessageUtilities.UpdateMessage(uxMessage, "The list of
Activity Types for publishing for a given user has been set.",
Message.DisplayModes.Success);
                        }
                    }
                    else
                    {
```

```
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Activity Type ID.", Message.DisplayModes.Error);
                        return;
                    }
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User ID and
Try again.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

AllowPublication

```
AllowPublication(System.Int64,System.Int64)
```

Retrieves true if the supplied activity type ID is enabled for publishing for the supplied User ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Activity Type ID

Parameters

- `userId`. ID to check publishing preferences for.
- `activityTypeId`. Activity Type ID to check publishing preferences for.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="*UserId :" />
```

```
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="10" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityTypeIdLabel"
AssociatedControlID="uxActivityTypeId"
            CssClass="span-4 last" runat="server" Text="*ActivityTypeId :" />
        <ektronUI:TextField ID="uxActivityTypeId" CssClass="span-6"
ValidationGroup="RegisterValidationGroup"
            runat="server" Text="20" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Allow"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPublishPreferenceManager notificationPublishPreferenceManager =
new NotificationPublishPreferenceManager();
            ActivityTypeManager activityTypeManager = new ActivityTypeManager();

            long userID = long.Parse(uxUserId.Text);
            long activityTypeID = long.Parse(uxActivityTypeId.Text);

            UserManager Usermanager = new UserManager();
            //Get the User data for given UserID
```

```
            UserData Userdata = Usermanager.GetItem(userID);
            //Check Whether Entered User ID is valid or Not.
            if (Userdata != null)
            {
                //Check whether Entered activityTypeID valid or Not
                ActivityTypeData activityTypeData = activityTypeManager.GetItem
(activityTypeID);
                if (activityTypeData != null)
                {
                    bool allow = notificationPublishPreferenceManager.AllowPublication
(userID, activityTypeID);
                    if (allow)
                    {
                        MessageUtilities.UpdateMessage(uxMessage,"Supplied
ActivityTypeId is enabled for publishing for the supplied User Id.",
Message.DisplayModes.Success);
                    }
                    else
                    {
                        MessageUtilities.UpdateMessage(uxMessage,  "Supplied
ActivityTypeId is NOT enabled for publishing for the supplied User Id.",
Message.DisplayModes.Error);
                    }
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Activity Type ID.", Message.DisplayModes.Error);
                    return;
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User ID and
Try again.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetDefaultList

```
GetDefaultList()
```

Retrieves the list of default notification publishing preferences for the system.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPublishPreferenceManager notificationPublishPreferenceManager =
new NotificationPublishPreferenceManager();

            List<NotificationPublishPreferenceData> notificationPublishPreferenceList =
notificationPublishPreferenceManager.GetDefaultList();

            uxnotificationPublishPreferenceListView.Visible = true;
            uxnotificationPublishPreferenceListView.DataSource =
notificationPublishPreferenceList;
            uxnotificationPublishPreferenceListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(System.Int64)
```

Retrieves the list of default notification publishing preferences for a user.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `userId`. User ID to return publishing preferences for.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="*UserId :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="10" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxnotificationPublishPreferenceListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        ActivityTypeId
                    </th>
                    <th>
                        ActivityTypeName
                    </th>
                    <th>
                        ActivityTypeScope
                    </th>
                    <th>
                        IsEnabled
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("ActivityTypeId")%>
```

```
            </td>
            <td class="devsite-method">
                <%# Eval("ActivityTypeName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ActivityTypeScope")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsEnabled")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPublishPreferenceManager notificationPublishPreferenceManager =
new NotificationPublishPreferenceManager();

            long userID = long.Parse(uxUserId.Text);

            UserManager Usermanager = new UserManager();
            //Get the User data for given UserID
            UserData Userdata = Usermanager.GetItem(userID);
            //Check Whether Entered User ID is valid or Not.
            if (Userdata != null)
            {
                List<NotificationPublishPreferenceData>
notificationPublishPreferenceList = notificationPublishPreferenceManager.GetList
(userID);

                uxnotificationPublishPreferenceListView.Visible = true;
                uxnotificationPublishPreferenceListView.DataSource =
notificationPublishPreferenceList;
                uxnotificationPublishPreferenceListView.DataBind();
            }
```

```
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User ID and
Try again.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

UpdateDefaultPreferences

```
UpdateDefaultPreferences(System.Collections,Generic.List)
```

Updates the list of activity types for publishing for a given user.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- *Activity Type IDs (comma separated)

Parameters

- `ActivityTypeIdList`. List of Activity Type IDs to enable publishing for.

Remarks

Validate the activity type IDs with `GetItem()` before you update the properties. Then, call `Update` with your modified ActivityTypeData object.

NOTE: You cannot change all properties after the initial Add event, such as the ActivityTypeData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxActivityTypeIdsLabel"
AssociatedControlID="uxActivityTypeIds"
            CssClass="span-4 last" runat="server" Text="*ActivityTypeIds (Comma
Separated):" />
        <ektronUI:TextField ID="uxActivityTypeIds" CssClass="span-6"
ValidationGroup="RegisterValidationGroup"
```

```
            runat="server" Text="0" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Activity;
using Ektron.Cms.Activity;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            NotificationPublishPreferenceManager notificationPublishPreferenceManager =
new NotificationPublishPreferenceManager();

            string[] activityTypeIds = uxActivityTypeIds.Text.Split(',', ';');
            List<long> activityTypeIdList = new List<long>();

            ActivityTypeManager activityTypeManager = new ActivityTypeManager();
            for (int i = 0; i < activityTypeIds.Length; i++)
            {
                long activityTypeID = long.Parse(activityTypeIds[i]);
                //Check whether Entered activityTypeID valid or Not
                ActivityTypeData activityTypeData = activityTypeManager.GetItem
(activityTypeID);
                if (activityTypeData != null)
                {
                    activityTypeIdList.Add(activityTypeID);
                    if (i == activityTypeIds.Length - 1)
                    {
                        notificationPublishPreferenceManager.UpdateDefaultPreferences
```

```
(activityTypeIdList);
                        MessageUtilities.UpdateMessage(uxMessage, "Default list of
Activity Types for publishing has been Updated.", Message.DisplayModes.Success);
                    }
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Activity Type ID.", Message.DisplayModes.Error);
                    return;
                }
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

UserNotificationSettingManager

8.50 and higher

The UserNotificationSettingManager class manages user-specified notifications. UserNotificationSettigns are user settings per notification agent. For example, to receive notifications via email, the user would need to provide his or her email address, and so on.

For information about notifications, see Sending Notifications to a Community in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Settings.Notifications.UserNotificationSettingManager
```

Constructors

- `UserNotificationSettingManager()`
- `UserNotificationSettingManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.
- `UserSettingService`. User setting service.

Methods

Add

```
Add(Ektron.Cms.UserNotificationSettingData)
```

Adds a UserNotificationSettingData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Agent ID
- Enable
- Verification Code

Parameters

- `userSettingData`. User setting data to save.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="*UserId :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAgentIdLabel" AssociatedControlID="uxAgentId"
CssClass="span-4 last"
            runat="server" Text="*AgentId :" />
        <ektronUI:TextField ID="uxAgentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="2" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnableLabel" AssociatedControlID="uxIsEnable"
CssClass="span-4 last"
            runat="server" Text="Enable :" />
        <asp:DropDownList ID="uxIsEnable" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxVerificationCodeLabel"
AssociatedControlID="uxVerificationCode"
            CssClass="span-4 last" runat="server" Text="Verification Code:" />
        <ektronUI:TextField ID="uxVerificationCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Notifications.Provider;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserNotificationSettingManager userNotificationSettingManager = new
UserNotificationSettingManager();
            bool status = false;
            if (uxIsEnable.SelectedValue.ToString() == "True")
                status = true;

            NotificationAgentSettingManager notificationAgentSettingManager = new
NotificationAgentSettingManager();
            long NotificationAgentID = long.Parse(uxAgentId.Text);
            //Check  Whether NotificationAgentID valid or Not
            NotificationAgentData notificationAgentData =
notificationAgentSettingManager.GetItem(NotificationAgentID);
            if (notificationAgentData != null)
            {
                UserManager Usermanager = new UserManager();
                long UserId = long.Parse(uxUserId.Text);
                //Check Whether Entered UserID Valid or Not
                UserData Userdata = Usermanager.GetItem(UserId);
                if (Userdata != null)
                {
```

```
                    NotificationAgent agent = NotificationAgentManager.Providers
[notificationAgentData.Name];
                    //Get the Agent Settings Data.
                    NotificationAgentSettingsData notificationAgentSettingsData =
agent.GetAgentSettings();
                    //Set User Notification Setting with given Details
                    UserNotificationSettingData userNotificationSettingData = new
UserNotificationSettingData()
                    {
                        AgentId = NotificationAgentID, //AgentId =2 is for 'Activity
Stream.'
                        UserId = long.Parse(uxUserId.Text),
                        IsEnabled = status,
                        VerificationCode = uxVerificationCode.Text != string.Empty ?
uxVerificationCode.Text : null,
                        AgentSettings = notificationAgentSettingsData,

                    };
                    userNotificationSettingManager.Add(userNotificationSettingData);

                    MessageUtilities.UpdateMessage(uxMessage, "User Notification Setting
Saved with Id " + userNotificationSettingData.Id, Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid User
Id and Try again.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid
NotificationAgent ID and try again", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a user notification setting from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Setting ID

Parameters

- `id`. ID of UserNotificationSettingData object to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserSettingIdLabel" AssociatedControlID="uxUserSettingId"
CssClass="span-4 last"
            runat="server" Text="*UserSetting Id :" />
        <ektronUI:TextField ID="uxUserSettingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserNotificationSettingManager userNotificationSettingManager = new
UserNotificationSettingManager();
            //Check whether Entered UserSettingId is Valid or Not
            long UserSettingId = long.Parse(uxUserSettingId.Text);
```

```
            UserNotificationSettingData userNotificationSettingData =
userNotificationSettingManager.GetItem(UserSettingId);
            if (userNotificationSettingData != null)
            {   //Delete a given User Notification Setting from the CMS.
                userNotificationSettingManager.Delete(UserSettingId);
                MessageUtilities.UpdateMessage(uxMessage, "User Notification Setting
with Id " + UserSettingId + " has been Deleted from the CMS",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter a Valid User
Notification Settings Id ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.UserNotificationSettingData)
```

Retrieves the properties of a specific UserNotificationSettingData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Setting ID

Parameters

- `id`. ID of UserNotificationSettingData object to return.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserSettingIdLabel" AssociatedControlID="uxUserSettingId"
CssClass="span-4 last"
            runat="server" Text="*UserSetting Id :" />
        <ektronUI:TextField ID="uxUserSettingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
```

```
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAgentID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxVerificationCode" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserNotificationSettingManager userNotificationSettingManager = new
UserNotificationSettingManager();
            //Check whether Entered UserSettingId is Valid or Not
            long UserSettingId = long.Parse(uxUserSettingId.Text);
            UserNotificationSettingData userNotificationSettingData =
userNotificationSettingManager.GetItem(UserSettingId);
            if (userNotificationSettingData != null)
            {    //Display the User Notification Setting Details.
                MessageUtilities.UpdateMessage(uxMessage, "User Notification Setting
with Id " + UserSettingId + " Details are below:", Message.DisplayModes.Success);

                uxAgentID.Text = "Agent Id : " + userNotificationSettingData.AgentId;
```

```
                uxUserId.Text = "User Id : " + userNotificationSettingData.UserId;
                uxIsEnabled.Text = "IsEnabled : " +
userNotificationSettingData.IsEnabled;
                uxVerificationCode.Text = "Verification Code : " +
userNotificationSettingData.VerificationCode;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter a Valid User
Notification Settings Id ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Notifications.UserNotificationSettingCriteria)
```

Retrieves lists of objects through the GetList(UserNotificatioSettingCriteria) method.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- User Notification Setting Property
- * Object Value

Parameters

- `criteria`. Criteria used to determine which UserNotificationSettingData to return.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserNotificationSettingPropertyLabel"
AssociatedControlID="uxUserNotificationSettingProperty"
            CssClass="span-5 last" runat="server" Text="UserNotification Setting
Property :" />
        <asp:DropDownList ID="uxUserNotificationSettingProperty" runat="server">
            <asp:ListItem>UserId</asp:ListItem>
```

```
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>AgentId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last"
            runat="server" Text="ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxNotificationSettingListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        UserId
                    </th>
                    <th>
                        AgentId
                    </th>
                    <th>
                        IsEnabled
                    </th>
                    <th>
                        VerificationCode
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
```

```
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("AgentId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsEnabled ")%>
            </td>
            <td class="devsite-method">
                <%# Eval("VerificationCode")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserNotificationSettingManager userNotificationSettingManager = new
UserNotificationSettingManager();
            object Objectvalue;
            string Property = uxUserNotificationSettingProperty.SelectedItem.Text;
            Objectvalue = Convert.ToInt64(uxObjectValue.Text);

            UserNotificationSettingCriteria criteria = new
UserNotificationSettingCriteria(UserNotificationSettingProperty.Id,
EkEnumeration.OrderByDirection.Ascending);
            if (Property == "Id")
            {
                criteria.AddFilter(UserNotificationSettingProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (Property == "UserId")
            {
                criteria.AddFilter(UserNotificationSettingProperty.UserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
```

```
            else
            {
                criteria.AddFilter(UserNotificationSettingProperty.AgentId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            List<UserNotificationSettingData> NotificationSettingList =
userNotificationSettingManager.GetList(criteria);

            uxNotificationSettingListView.Visible = true;
            uxNotificationSettingListView.DataSource = NotificationSettingList;
            uxNotificationSettingListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Common.CustomPropertyData)
```

Updates an existing UserNotificationSettingData object item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Setting ID
- User ID
- Agent ID
- Enable
- Verification Code

Parameters

- `userSettingData`. User setting data to save.

Returns

Returns the custom CmsData object updated.

Remarks

Validate the user notification setting item with `GetItem()` before you update the properties. Then, call `Update` with your modified UserNotificationSettingData object.

NOTE: You cannot change all properties after the initial Add event, such as the UserNotificationSettingData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserSettingIdLabel" AssociatedControlID="uxUserSettingId"
CssClass="span-4 last"
            runat="server" Text="*UserSetting Id :" />
        <ektronUI:TextField ID="uxUserSettingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="UserId :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAgentIdLabel" AssociatedControlID="uxAgentId"
CssClass="span-4 last"
            runat="server" Text="AgentId :" />
        <ektronUI:TextField ID="uxAgentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnableLabel" AssociatedControlID="uxIsEnable"
CssClass="span-4 last"
            runat="server" Text="Enable :" />
        <asp:DropDownList ID="uxIsEnable" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxVerificationCodeLabel"
AssociatedControlID="uxVerificationCode"
            CssClass="span-4 last" runat="server" Text="Verification Code:" />
        <ektronUI:TextField ID="uxVerificationCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
```

```
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Notifications.Provider;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            if (uxIsEnable.SelectedValue.ToString() == "True")
                status = true;

            //Check whether Entered UserSettingId is Valid or Not
            long UserSettingId = long.Parse(uxUserSettingId.Text);
            userNotificationSettingData = userNotificationSettingManager.GetItem
(UserSettingId);
            if (userNotificationSettingData != null)
            {
                UserManager Usermanager = new UserManager();
                long UserId = long.Parse(uxUserId.Text);
                if (UserId > 0)
                {
                    //Check Whether Entered UserID Valid or Not
                    UserData Userdata = Usermanager.GetItem(UserId);
                    if (Userdata != null)
                    {    //Update USER ID here
                        userNotificationSettingData.UserId = UserId;
                        update();
                    }
                    else
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid
User Id and Try again.", Message.DisplayModes.Error);
                    }
                }
                else { update(); }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter a Valid User
```

```
Notification Settings Id ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

VerifyValidationCode

```
VerifyValidationCode(System.Int64,System.String)
```

Verifies that the validation code is correct and enables users notfication settings.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Setting ID
- * Verification Code

Parameters

- `userSettingId`. ID of user notification setting.
- `verificationCode`. Verification code to validate.

Returns

Returns true if the verification code is correct.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserSettingIdLabel" AssociatedControlID="uxUserSettingId"
CssClass="span-4 last"
            runat="server" Text="*UserSetting Id :" />
        <ektronUI:TextField ID="uxUserSettingId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxVerificationCodeLabel"
AssociatedControlID="uxVerificationCode"
            CssClass="span-4 last" runat="server" Text="*Verification Code:" />
        <ektronUI:TextField ID="uxVerificationCode" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
```

```
            Text="" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

    <ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.Notifications;
using Ektron.Cms.Notifications;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserNotificationSettingManager userNotificationSettingManager = new
UserNotificationSettingManager();
            //Check whether Entered UserSettingId is Valid or Not
            long UserSettingId = long.Parse(uxUserSettingId.Text);
            UserNotificationSettingData userNotificationSettingData =
userNotificationSettingManager.GetItem(UserSettingId);
            if (userNotificationSettingData != null)
            {
                bool status = userNotificationSettingManager.VerifyValidationCode
(UserSettingId, uxVerificationCode.Text);
                if (status == true)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Entered verification code
is correct.", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Entered verification code
is Not correct ", Message.DisplayModes.Error);
                }
            }
            else
            {
```

```
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter a Valid User
Notification Settings Id ", Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

UserNotificationSettingCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Notifications
```

Constructors

- `UserNotificationSettingCriteria()`

  ```
  public UserNotificationSettingCriteria()
  ```

- `UserNotificationSettingCriteria (Ektron.Cms.Notifications.UserNotificationSettingProperty, EkEnumeration.OrderByDirection)`

  ```
  public
    UserNotificationSettingCriteria
  (Ektron.Cms.Notifications.UserNotificationSettingProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `UserNotificationSettingProperty` are:

  - AgentId
  - Id
  - IsEnabled
  - UserId
  - VerificationCode

UserNotificationSettingData

Namespace

```
Ektron.Cms.Notifications
```

Properties

- `AgentId`. Gets or sets the ID of the Notification Agent associated with these settings.

```
public long AgentId { set; get; }
```

- `AgentSettings`. Gets or sets the user's settings for this notification provider.

```
public Ektron.Cms.Notifications.NotificationAgentSettingsData
  AgentSettings { set; get; }
```

- `IsEnabled`. Gets or sets the enabled setting for this user notification setting. This notification setting cannot be used if its not enabled.

```
public bool IsEnabled { set; get; }
```

- `UserId`. Gets or sets the user Id associated with these settings.

```
public long UserId { set; get; }
```

- `UserNotificationSettingData()`

```
public UserNotificationSettingData()
```

- `UserNotificationSettingData(long, long, long)`

```
public UserNotificationSettingData(long id, long userId,
  long agentId)
```

- `UserNotificationSettingData(long, long, long, Ektron.Cms.Notifications.NotificationAgentSettingsData)`

```
public UserNotificationSettingData(long id, long userId,
  long agentId, Ektron.Cms.Notifications.NotificationAgentSettingsData
  agentSettings)
```

- `Validate()`. Validates that the data object is valid for saving. Any errors are returns as ValidationResults.

```
public ValidationResults Validate()
```

- `VerificationCode`. Gets or sets the verificationcode for this setting. The user cannot recieve notifications on this provider until they verify this code.

```
public string VerificationCode { set; get; }
```

PermissionManager

8.50 and higher

The PermissionManager class manages user permissions. See *Managing Folder and Content Permissions* in Setting Up Your CMS Folder Structure in the Ektron Reference for information about standard and advanced permissions.

Namespace

```
Ektron.Cms.Framework.Settings.PermissionManager
```

Constructors

- `PermissionManager()`
- `PermissionManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `PermissionManagerService`. Returns an instance of the business objects PermissionManager service.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- `GetUserPermissionForContent(Int64,Int64,Int32).` 8.60 and higher Gets the users permission to a piece of content, for either the user or group the user is part of.
- `GetUserPermissionForFolder(Int64,Int64,Int32).` 8.60 and higher Gets the users permission to a folder, for either the user or group the user is part of.
- `IsLoggedInUserAdmin().` 8.60 and higher Checks if currently logged in user is a CMS administrator
- *Update* on page 1570

Add

```
Add(Ektron.Cms.Messaging.UserPermissionData)
```

Adds a user permission, with details from the supplied UserPermissionData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Content ID
- * User ID
- Is Membership User
- Is Inherited
- Is Private Content
- Is Read Only
- Can Edit
- Can Add
- Can Delete
- Is Read Only Lib

Parameters

- `permissionData`

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
```

```
CssClass="span-4 last"
            runat="server" Text="*Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="30" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="10" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsMembershipUserLabel"
AssociatedControlID="uxIsMembershipUser"
            CssClass="span-4 last" runat="server" Text="IsMembershipUser :" />
        <asp:DropDownList ID="uxIsMembershipUser" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsInheritedLabel" AssociatedControlID="uxIsInherited"
CssClass="span-4 last"
            runat="server" Text="IsInherited :" />
        <asp:DropDownList ID="uxIsInherited" runat="server" CssClass="span-4 last">
            <asp:ListItem>False</asp:ListItem>
            <asp:ListItem>True</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsPrivateContentLabel"
AssociatedControlID="uxIsPrivateContent"
            CssClass="span-4 last" runat="server" Text="IsPrivateContent :" />
        <asp:DropDownList ID="uxIsPrivateContent" runat="server" CssClass="span-4 last">
            <asp:ListItem>False</asp:ListItem>
            <asp:ListItem>True</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsReadOnlyLabel" AssociatedControlID="uxIsReadOnly"
CssClass="span-4 last"
            runat="server" Text="IsReadOnly :" />
        <asp:DropDownList ID="uxIsReadOnly" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCanEditLabel" AssociatedControlID="uxCanEdit"
CssClass="span-4 last"
            runat="server" Text="CanEdit :" />
        <asp:DropDownList ID="uxCanEdit" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
```

```
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCanAddLabel" AssociatedControlID="uxCanAdd"
CssClass="span-4 last"
            runat="server" Text="CanAdd :" />
        <asp:DropDownList ID="uxCanAdd" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCanDeleteLabel" AssociatedControlID="uxCanDelete"
CssClass="span-4 last"
            runat="server" Text="CanDelete :" />
        <asp:DropDownList ID="uxCanDelete" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsReadOnlyLibLabel" AssociatedControlID="uxIsReadOnlyLib"
CssClass="span-4 last"
            runat="server" Text="IsReadOnlyLib :" />
        <asp:DropDownList ID="uxIsReadOnlyLib" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Framework.User;
```

```
using Microsoft.VisualBasic;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PermissionManager permissionManager = new PermissionManager();
            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxUserId.Text);
            //Check whether entered UserID is Valid or Not
            UserData Userdata = Usermanager.GetItem(UserId);
            if (Userdata != null && Userdata.IsDeleted == false)
            {
                ContentManager contentManager = new ContentManager();
                long contentID = long.Parse(uxContentId.Text);
                bool result = Information.IsNumeric(uxContentId.Text);
                ContentData contentData = contentManager.GetItem(contentID);
                //Check whether entered ContentID is Valid or Not
                if (!((result) && contentData != null))
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid content
ID ", Message.DisplayModes.Error);
                    return;
                }
                else
                {
                    //Set the UserPermission Details
                    UserPermissionData userPermissionData = new UserPermissionData()
                    {
                        ContentId = contentID,
                        FolderId = contentData.FolderId,
                        UserId = UserId,
                        IsInherited = bool.Parse(uxIsInherited.SelectedValue),
                        IsPrivateContent = bool.Parse(uxIsPrivateContent.SelectedValue),
                        IsReadOnly = bool.Parse(uxIsReadOnly.SelectedValue),
                        CanEdit = bool.Parse(uxCanEdit.SelectedValue),
                        CanAdd = bool.Parse(uxCanAdd.SelectedValue),
                        CanDelete = bool.Parse(uxCanDelete.SelectedValue),
                        IsReadOnlyLib = bool.Parse(uxIsReadOnlyLib.SelectedValue),
                        IsMembershipUser = bool.Parse(uxIsMembershipUser.SelectedValue),
                    };
                    //Add a New UserPermissionData with Given Details
                    permissionManager.Add(userPermissionData);
                    MessageUtilities.UpdateMessage(uxMessage, "User Permission Saved
with Id " + userPermissionData.Id, Message.DisplayModes.Success);
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User ID and
Try again.", Message.DisplayModes.Error);
```

```
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Delete

```
Delete(System.Int64)
```

Deletes a user permission from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Permission ID

Parameters

- `id`. User or Group ID.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserPermissionIdLabel"
AssociatedControlID="uxUserPermissionId"
            CssClass="span-4 last" runat="server" Text="*UserPermission Id:" />
        <ektronUI:TextField ID="uxUserPermissionId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
```

```
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Framework.User;
using Microsoft.VisualBasic;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PermissionManager permissionManager = new PermissionManager();
            UserPermissionData userPermissionData = new UserPermissionData();
            long UserPermissionId = long.Parse(uxUserPermissionId.Text);
            userPermissionData = permissionManager.GetItem(UserPermissionId);
            if (userPermissionData != null)
            {
                permissionManager.Delete(UserPermissionId);
                MessageUtilities.UpdateMessage(uxMessage, "UserPermission with Id " +
UserPermissionId + " has been deleted from the CMS.", Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User
Permission ID", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

DeletePermissionForGroup

```
DeletePermissionForGroup(Ektron.Cms.Settings.ItemType,
System.Int64,System.Int64)
```

Deletes an object permission from the CMS based on type and group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Group ID
- * Item Type
- * Item ID

Parameters

- `type`. Type of the object to delete associated permission.
- `groupId`. ID of the group to which delete associated permissions.
- `objectId`. ID of the object to delete permissions.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxgroupIDLabel" AssociatedControlID="uxgroupID"
            CssClass="span-4 last" runat="server" Text="*Group Id:" />
        <ektronUI:TextField ID="uxgroupID" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxItemTypeLabel" AssociatedControlID="uxItemType"
            CssClass="span-4 last" runat="server" Text="*ItemType :" />
        <asp:DropDownList ID="uxItemType" runat="server" CssClass="span-4">
            <asp:ListItem>Folder</asp:ListItem>
            <asp:ListItem>Content</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxItemIdLabel" AssociatedControlID="uxItemId"
            CssClass="span-4 last" runat="server" Text="*Item Id:" />
```

```
        <ektronUI:TextField ID="uxItemId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Framework.User;
using Microsoft.VisualBasic;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PermissionManager permissionManager = new PermissionManager();
            long groupID = long.Parse(uxgroupID.Text);
            long itemId = long.Parse(uxItemId.Text);
            string itemType = uxItemType.SelectedItem.Text;

            UserGroupManager userGroupManager = new UserGroupManager();
            //Check Entered GroupID is Valid or Not
            UserGroupData userGroupData = userGroupManager.GetItem(groupID);
            if (userGroupData != null && userGroupData.Id > 0)
            {
                if (itemType == "Folder")
                {
                    FolderManager folderManager = new FolderManager();
```

```
                    FolderData folderData = folderManager.GetItem(itemId, false);
                    //Check Entered FolderId is Valid or Not
                    if (folderData != null)
                    {
                        permissionManager.DeletePermissionForGroup
(Ektron.Cms.Settings.ItemType.Folder, groupID, itemId);
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus,  "Please Enter Valid
Folder ID.", Message.DisplayModes.Error);
                        return;
                    }
                }
                else
                {
                    ContentManager contentManager = new ContentManager();
                    ContentData cData = contentManager.GetItem(itemId, false);
                    //Check Entered ContentID is Valid or Not
                    if (cData != null && cData.Id > 0)
                    {
                        permissionManager.DeletePermissionForGroup
(Ektron.Cms.Settings.ItemType.Content, groupID, itemId);
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Content ID.", Message.DisplayModes.Error);
                        return;
                    }
                }

            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User ID.",
Message.DisplayModes.Error);
                return;
            }

            MessageUtilities.UpdateMessage(uxMessage, "Permissions for an Object has
been Deleted.", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeletePermissionsForUser

```
DeletePermissionsForUser(Ektron.Cms.Settings.ItemType,System.Int64,System.Int64)
```

Deletes an object permission from the CMS based on type and user.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Item Type
- * Item ID

Parameters

- `type`. Type of the object to delete associated permission.
- `userId`. User ID of the user to which delete associated permissions.
- `objectId`. ID of the object to delete permissions.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
            CssClass="span-4 last" runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxItemTypeLabel" AssociatedControlID="uxItemType"
            CssClass="span-4 last" runat="server" Text="*ItemType :" />
        <asp:DropDownList ID="uxItemType" runat="server" CssClass="span-4">
            <asp:ListItem>Folder</asp:ListItem>
            <asp:ListItem>Content</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxItemIdLabel" AssociatedControlID="uxItemId"
            CssClass="span-4 last" runat="server" Text="*Item Id:" />
        <ektronUI:TextField ID="uxItemId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
```

```
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Framework.User;
using Microsoft.VisualBasic;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PermissionManager permissionManager = new PermissionManager();
            long userID = long.Parse(uxUserId.Text);
            long itemId = long.Parse(uxItemId.Text);
            string itemType = uxItemType.SelectedItem.Text;

            UserManager Usermanager = new UserManager();
            //Check Entered UserID is Valid or Not
            UserData Userdata = Usermanager.GetItem(userID);
            if (Userdata != null && Userdata.Id > 0)
            {
                if (itemType == "Folder")
                {
                    FolderManager folderManager = new FolderManager();
                    FolderData folderData = folderManager.GetItem(itemId, false);
                    //Check Entered FolderId is Valid or Not
                    if (folderData != null)
                    {
                        permissionManager.DeletePermissionsForUser
(Ektron.Cms.Settings.ItemType.Folder, userID, itemId);
```

```
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Folder ID.", Message.DisplayModes.Error);
                        return;
                    }
                }
                else
                {
                    ContentManager contentManager = new ContentManager();
                    ContentData cData = contentManager.GetItem(itemId, false);
                    //Check Entered ContentID is Valid or Not
                    if (cData != null && cData.Id > 0)
                    {
                        permissionManager.DeletePermissionsForUser
(Ektron.Cms.Settings.ItemType.Content, userID, itemId);
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Content ID.", Message.DisplayModes.Error);
                        return;
                    }
                }

            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User ID.",
Message.DisplayModes.Error);
                return;
            }

            MessageUtilities.UpdateMessage(uxMessage, "Permissions for an Object has
been Deleted.", Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.UserPermissionData)
```

Retrieves the properties of a specific UserPermissionData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Permission ID

Parameters

- `id`. ID of the UserPermission to get.

Returns

UserPermission details in a UserPermissionData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserPermissionIdLabel"
AssociatedControlID="uxUserPermissionId"
            CssClass="span-4 last" runat="server" Text="*UserPermission Id:" />
        <ektronUI:TextField ID="uxUserPermissionId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsInherited" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
```

```
        <asp:Literal ID="uxIsPrivateContent" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsReadOnly" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCanEdit" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCanAdd" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCanDelete" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsReadOnlyLib" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Framework.User;
using Microsoft.VisualBasic;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PermissionManager permissionManager = new PermissionManager();
            UserPermissionData userPermissionData = new UserPermissionData();
            userPermissionData = permissionManager.GetItem(long.Parse
(uxUserPermissionId.Text));
            if (userPermissionData != null)
            {
                //Display the userPermission Details
                MessageUtilities.UpdateMessage(uxMessage, "UserPermission with Id " +
userPermissionData.Id + " details are below:", Message.DisplayModes.Success);

                uxUserID.Text = "UserId : " + userPermissionData.UserId;
                uxContentId.Text = "Content Id : " + userPermissionData.ContentId;
                uxIsInherited.Text = "IsInherited : " + userPermissionData.IsInherited;
                uxIsPrivateContent.Text = "IsPrivateContent : " +
userPermissionData.IsPrivateContent;
                uxIsReadOnly.Text = "IsReadOnly : " + userPermissionData.IsReadOnly;
```

```
                uxCanEdit.Text = "CanEdit : " + userPermissionData.CanEdit;
                uxCanAdd.Text = "CanAdd : " + userPermissionData.CanAdd;
                uxCanDelete.Text = "CanDelete : " + userPermissionData.IsPrivateContent;
                uxIsReadOnlyLib.Text = "IsReadOnlyLib : " +
userPermissionData.IsReadOnlyLib;
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User
Permission ID", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Settings.PermissionCriteria)
```

Retrieves lists of objects through the GetList(PermissionCriteria) method.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- User Permission Property
- * Object Value

Parameters

- `criteria`. PermissionCriteria used to specify or filter the UserPermissionData object to return.

Returns

Returns a list of UserPermissions.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxUserPermissionPropertyLabel"
AssociatedControlID="uxUserPermissionProperty"
            CssClass="span-6 last" runat="server" Text="UserPermission Property :" />
        <asp:DropDownList ID="uxUserPermissionProperty" runat="server">
            <asp:ListItem>UserId</asp:ListItem>
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>ContentId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last"
            runat="server" Text="ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxUserPermissionListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        UserId
                    </th>
                    <th>
                        FolderId
                    </th>
                    <th>
                        ContentId
                    </th>
                    <th>
                        IsInherited
                    </th>
                    <th>
                        IsReadOnly
                    </th>
                    <th>
                        IsReadOnlyLib
                    </th>
                    <th>
                        IsPrivateContent
```

```
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FolderId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ContentId ")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsInherited")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsReadOnly")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsReadOnlyLib")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsPrivateContent")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Framework.User;
using Microsoft.VisualBasic;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Settings;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            PermissionManager permissionManager = new PermissionManager();
            object Objectvalue;
            string Property = uxUserPermissionProperty.SelectedItem.Text;
            Objectvalue = Convert.ToInt64(uxObjectValue.Text);

            PermissionCriteria criteria = new PermissionCriteria();
            criteria.OrderByDirection = EkEnumeration.OrderByDirection.Ascending;
            criteria.OrderByField = PermissionProperty.Id;

            if (Property == "Id")
            {
                criteria.AddFilter(PermissionProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else if (Property == "UserId")
            {
                criteria.AddFilter(PermissionProperty.UserId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                criteria.AddFilter(PermissionProperty.ContentId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            List<UserPermissionData> UserPermissionList = permissionManager.GetList
(criteria);
            uxUserPermissionListView.Visible = true;
            uxUserPermissionListView.DataSource = UserPermissionList;
            uxUserPermissionListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.UserPermissionData)
```

Updates an existing UserPermissionData item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Permission ID
- Content ID
- User ID
- Is Membership User
- Is Inherited
- Is Private Content
- Is Read Only
- Can Edit
- Can Add
- Can Delete
- Is Read Only Lib

Parameters

- `permissionData`

Returns

Returns the UserPermissionData object that was updated.

Remarks

Validate the User Permission Data item with `GetItem()` before you update the properties. Then, call `Update` with your modified UserPermissionData object.

NOTE: You cannot change all properties after the initial Add event, such as the UserPermissionData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserPermissionIdLabel"
AssociatedControlID="uxUserPermissionId"
            CssClass="span-4 last" runat="server" Text="*UserPermission Id:" />
        <ektronUI:TextField ID="uxUserPermissionId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-4 last"
            runat="server" Text="Content Id:" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
```

```
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-4 last"
            runat="server" Text="User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsMembershipUserLabel"
AssociatedControlID="uxIsMembershipUser"
            CssClass="span-4 last" runat="server" Text="IsMembershipUser :" />
        <asp:DropDownList ID="uxIsMembershipUser" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsInheritedLabel" AssociatedControlID="uxIsInherited"
CssClass="span-4 last"
            runat="server" Text="IsInherited :" />
        <asp:DropDownList ID="uxIsInherited" runat="server" CssClass="span-4 last">
            <asp:ListItem>False</asp:ListItem>
            <asp:ListItem>True</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsPrivateContentLabel"
AssociatedControlID="uxIsPrivateContent"
            CssClass="span-4 last" runat="server" Text="IsPrivateContent :" />
        <asp:DropDownList ID="uxIsPrivateContent" runat="server" CssClass="span-4 last">
            <asp:ListItem>False</asp:ListItem>
            <asp:ListItem>True</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsReadOnlyLabel" AssociatedControlID="uxIsReadOnly"
CssClass="span-4 last"
            runat="server" Text="IsReadOnly :" />
        <asp:DropDownList ID="uxIsReadOnly" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCanEditLabel" AssociatedControlID="uxCanEdit"
CssClass="span-4 last"
            runat="server" Text="CanEdit :" />
        <asp:DropDownList ID="uxCanEdit" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxCanAddLabel" AssociatedControlID="uxCanAdd"
CssClass="span-4 last"
            runat="server" Text="CanAdd :" />
        <asp:DropDownList ID="uxCanAdd" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxCanDeleteLabel" AssociatedControlID="uxCanDelete"
CssClass="span-4 last"
            runat="server" Text="CanDelete :" />
        <asp:DropDownList ID="uxCanDelete" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsReadOnlyLibLabel" AssociatedControlID="uxIsReadOnlyLib"
CssClass="span-4 last"
            runat="server" Text="IsReadOnlyLib :" />
        <asp:DropDownList ID="uxIsReadOnlyLib" runat="server" CssClass="span-4 last">
            <asp:ListItem>True</asp:ListItem>
            <asp:ListItem>False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Framework.User;
using Microsoft.VisualBasic;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            userPermissionData = permissionManager.GetItem(long.Parse
(uxUserPermissionId.Text));
            if (userPermissionData != null)
            {
                UserManager Usermanager = new UserManager();
                UserId = long.Parse(uxUserId.Text);
                if (UserId > 0)
                {
                    //Check whether entered UserID is Valid or Not
                    UserData Userdata = Usermanager.GetItem(UserId);
                    if (Userdata != null && Userdata.IsDeleted == false)
                    {
                        userPermissionData.UserId = UserId;
                        update();
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
User ID", Message.DisplayModes.Error);
                        return;
                    }
                }
                else
                {
                    update();
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User
Permission ID", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Data Classes

PermissionCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Settings
```

Constructors

- `PermissionCriteria()`

  ```
  public PermissionCriteria()
  ```

- `PermissionCriteria(Ektron.Cms.Settings.PermissionProperty, EkEnumeration.OrderByDirection)`

  ```
  public PermissionCriteria(Ektron.Cms.Settings.PermissionProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `PermissionProperty` are:

  - CanAdd
  - CanDelete
  - CanEdit
  - ContentId
  - FolderId
  - Id
  - IsReadOnly
  - LanguageId
  - UserGroupId
  - UserId

UserPermissionData

Namespace

```
Ektron.Cms
```

Properties

- `ContentId`

  ```
  public long ContentId { set; get; }
  ```

- `DisplayGroupName`

  ```
  public string DisplayGroupName { set; get; }
  ```

- `DisplayUserName`

  ```
  public string DisplayUserName { set; get; }
  ```

- `Domain`

  ```
  public string Domain { set; get; }
  ```

- `FolderId`

  ```
  public long FolderId { set; get; }
  ```

- GroupDomain

```
public string GroupDomain { set; get; }
```

- GroupId

```
public long GroupId { set; get; }
```

- GroupName

```
public string GroupName { set; get; }
```

- InheritedFrom

```
public long InheritedFrom { set; get; }
```

- IsInherited

```
public bool IsInherited { set; get; }
```

- IsMembershipUser

```
public bool IsMembershipUser { set; get; }
```

- IsPrivateContent

```
public bool IsPrivateContent { set; get; }
```

- UserId

```
public long UserId { set; get; }
```

- UserName

```
public string UserName { set; get; }
```

- Validate()

```
public override
  Microsoft.Practices.EnterpriseLibrary.Validation.ValidationResults
  Validate()
```

SmartformConfigurationManager

8.50 and higher

The SmartformConfigurationManager class manages Smart Form configurations. See Working with Smart Forms in the Ektron Reference for information about Smart Forms.

Namespace

```
Ektron.Cms.Framework.Settings.SmartFormConfigurationManager
```

Constructors

- `SmartFormConfigurationManager()`
- `SmartFormConfigurationManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `SmartFormConfigurationManagerService`. Returns an instance of the business objects smartFormConfiguration manager service.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* on the next page
- *Delete* on page 1581
- *GetItem* on page 1583
- *GetList* on page 1585
- `GetXmlSearchFieldData(Int64)`. 8.60 and higher Retrieves a list of SmartFormIndexFieldData objects for the supplied smart form config id.
- *Update* on page 1588

Add

```
Add(Ektron.Cms.SmartFormConfigurationData)
```

Adds a SmartForm configuration based on information in an SmartFormConfigurationData object.

The method populates the message.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- * Type
- Description
- Field List

Parameters

- `smartFormConfigurationData`. The SmartFormConfigurationData object to add.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-3
last"
            runat="server" Text="* Title:" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTypeLabel" AssociatedControlID="uxType" CssClass="span-3
last"
            runat="server" Text="* Type:" />
        <asp:DropDownList ID="uxType" runat="server">
            <asp:ListItem Value="0">Content</asp:ListItem>
            <asp:ListItem Value="1">Product</asp:ListItem>
            <asp:ListItem Value="2">WebEvent</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
```

```
CssClass="span-3 last"
            runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3" ID="uxDescription" CssClass="span-4"
runat="server"
            ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Framework.User;
using Microsoft.VisualBasic;
using Ektron.Cms.Framework.Content;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ContentAPI _ContentApi = new ContentAPI();
            SmartFormConfigurationManager smartFormConfigurationManager = new
SmartFormConfigurationManager();
            UserManager userManager = new UserManager();
            if (userManager.UserId > 0)
            {
                SmartFormConfigurationData smartFormConfigurationData = new
SmartFormConfigurationData()
                {
                    SmartformTitle = uxTitle.Text,
                    SmartformDescription = uxDescription.Text,
                    Type = (EkEnumeration.XmlConfigType)(Convert.ToInt32
(uxType.SelectedItem.Value)),
                };
```

```
                smartFormConfigurationManager.Add(smartFormConfigurationData);
                smartFormConfigurationData = smartFormConfigurationManager.GetItem
(smartFormConfigurationData.Id);

                if (uxFieldList.Checked)
                {
                    string strcontent = "";
                    string strDesign = "Name: <input type=\"text\" class=\"design_
textfield\" size=\"24\" ektdesignns_nodetype=\"element\" ektdesignns_indexed=\"false\"
title=\"FullName\" ektdesignns_name=\"FullName\" ektdesignns_caption=\"FullName\"
id=\"FullName\" name=\"FullName\" /><p> </p>";

                    string strSchema = _ContentApi.TransformXsltPackage(strDesign,
Server.MapPath("../../../../Workarea/ContentDesigner/DesignToSchema.xslt"), true);
                    string strFieldList = _ContentApi.TransformXsltPackage(strDesign,
Server.MapPath("../../../../Workarea/ContentDesigner/DesignToFieldList.xslt"), true);
                    string strViewEntryXslt = _ContentApi.TransformXsltPackage
(strDesign, Server.MapPath
("../../../../Workarea/ContentDesigner/DesignToEntryXSLT.xslt"), true);

                    System.Xml.Xsl.XsltArgumentList objXsltArgs = new
System.Xml.Xsl.XsltArgumentList();
                    objXsltArgs.AddParam("srcPath", "",
"../../../../Workarea/ContentDesigner/");

                    string strViewXslt = _ContentApi.XSLTransform("<root>" + strDesign +
"<ektdesignpackage_list>" + strFieldList + "</ektdesignpackage_list></root>",
Server.MapPath("../../../../Workarea/ContentDesigner/DesignToViewXSLT.xslt"), true,
false, objXsltArgs, false);
                    string strInitDoc = _ContentApi.TransformXsltPackage(strDesign,
Server.MapPath("../../../../Workarea/ContentDesigner/PresentationToData.xslt"), true);

                    StringBuilder sbPackage = new StringBuilder();
                    sbPackage.Append("<ektdesignpackage_forms><ektdesignpackage_
form><ektdesignpackage_designs><ektdesignpackage_design>");
                    sbPackage.Append(strDesign);
                    sbPackage.Append("</ektdesignpackage_design></ektdesignpackage_
designs><ektdesignpackage_schemas><ektdesignpackage_schema>");
                    sbPackage.Append(strSchema);
                    sbPackage.Append("</ektdesignpackage_schema></ektdesignpackage_
schemas><ektdesignpackage_lists><ektdesignpackage_list>");
                    sbPackage.Append(strFieldList);
                    sbPackage.Append("</ektdesignpackage_list></ektdesignpackage_
lists><ektdesignpackage_views><ektdesignpackage_view>");
                    sbPackage.Append(strViewEntryXslt);
                    sbPackage.Append("</ektdesignpackage_view><ektdesignpackage_view>");
                    sbPackage.Append(strViewXslt);
                    sbPackage.Append("</ektdesignpackage_view></ektdesignpackage_
views><ektdesignpackage_initialDocuments><ektdesignpackage_initialDocument>");
                    sbPackage.Append(strInitDoc);
                    sbPackage.Append("</ektdesignpackage_
initialDocument></ektdesignpackage_initialDocuments></ektdesignpackage_
form></ektdesignpackage_forms>");
                    strcontent = sbPackage.ToString();
```

```
                    string _PackDisplayXslt = "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>" + "<xsl:stylesheet version=\"1.0\"
xmlns:xsl=\"http://www.w3.org/1999/XSL/Transform\">" + "<xsl:output method=\"xml\"
version=\"1.0\" encoding=\"UTF-8\" indent=\"yes\"/>" + "<xsl:template match=\"/\">" +
"<xsl:choose>" + "<xsl:when test=\"ektdesignpackage_forms/ektdesignpackage_form
[1]/ektdesignpackage_views/ektdesignpackage_view[2]\">" + "<xsl:copy-of
select=\"ektdesignpackage_forms/ektdesignpackage_form           [1]/ektdesignpackage_
views/ektdesignpackage_view[2]/node()\"/>" + "</xsl:when>" + "  <xsl:otherwise>" +
"<xsl:copy-of select=\"ektdesignpackage_forms/ektdesignpackage_form
[1]/ektdesignpackage_views/ektdesignpackage_view[1]/node()\"/>" + "</xsl:otherwise>" +
"</xsl:choose>" + "</xsl:template>" + "</xsl:stylesheet>";
                    string displayXslt = _ContentApi.TransformXsltPackage
(strcontent.ToString(), _PackDisplayXslt, false);

                    smartFormConfigurationData.PackageXslt = strcontent;
                    smartFormConfigurationData.PackageDisplayXslt = displayXslt;
                    smartFormConfigurationData.FieldList = strFieldList;

                    smartFormConfigurationManager.Update(smartFormConfigurationData);
                }


                MessageUtilities.UpdateMessage(uxMessage, "Smartform Configuration added
with Id " + smartFormConfigurationData.Id, Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Log In.",
Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a SmartFormConfigurationData object from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the SmartFormConfigurationData object to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxSmartFormIdLabel" AssociatedControlID="uxSmartFormId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxSmartFormId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
        </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long smartFormId = long.Parse(uxSmartFormId.Text);

            SmartFormConfigurationManager smartFormConfigurationManager = new
SmartFormConfigurationManager();
            SmartFormConfigurationData smartFormConfigurationData =
smartFormConfigurationManager.GetItem(smartFormId);
```

```
            if (smartFormConfigurationData != null)
            {
                smartFormConfigurationManager.Delete(smartFormId);
                MessageUtilities.UpdateMessage(uxMessage,  "Smartform deleted.",
Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid SmartForm
ID.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.SmartFormConfigurationData)
```

Retrieves the properties of a specific SmartFormConfigurationData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the SmartFormConfigurationData object to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxSmartFormIdLabel" AssociatedControlID="uxSmartFormId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxSmartFormId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
```

```
runat="server"
                    Text="Status:" />
        </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDescription" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDefaultXslt" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long smartFormId = long.Parse(uxSmartFormId.Text);

            SmartFormConfigurationManager smartFormConfigurationManager = new
SmartFormConfigurationManager();
            SmartFormConfigurationData smartFormConfigurationData =
smartFormConfigurationManager.GetItem(smartFormId);

            if (smartFormConfigurationData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for smart form with
```

```
ID " + smartFormConfigurationData.Id.ToString() + " are below",
Message.DisplayModes.Success);

                uxTitle.Text = "Title : " + smartFormConfigurationData.SmartformTitle ;
                uxDescription.Text = "Description : " +
smartFormConfigurationData.SmartformDescription;
                uxDefaultXslt.Text = "Default Xslt : " +
smartFormConfigurationData.DefaultXslt;
                uxType.Text = "Type : " + smartFormConfigurationData.Type;
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid SmartForm
ID.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Content.SmartFormConfigurationCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Smart Form Configuration Data Property
- * Object Value

Parameters

- `criteria`. SmartFormConfigurationCriteria used to retrieve SmartFormConfigurationData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxSmartFormConfigurationDataPropertyLabel"
AssociatedControlID="uxCmsMessageProperty" CssClass="span-6 last" runat="server"
Text="SmartFormConfigurationDataProperty:" />
        <asp:DropDownList ID="uxSmartFormConfigurationDataProperty" runat="server">
            <asp:ListItem>Title</asp:ListItem>
            <asp:ListItem>Type(0,1,2)</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxSmartFormConfigurationDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
                        Type
                    </th>
                    <th>
                        Default Xslt
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
```

```
            <td class="devsite-method">
                <%# Eval("SmartformTitle")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Type")%>
            </td>
            <td class="devsite-method">
                <%# Eval("DefaultXslt")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Content ;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            SmartFormConfigurationManager smartFormConfigurationManager = new
SmartFormConfigurationManager();

            SmartFormConfigurationCriteria criteria = new SmartFormConfigurationCriteria
();
            if (uxSmartFormConfigurationDataProperty.SelectedItem.Text == "Type(0,1,2)")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(SmartFormConfigurationProperty.Type,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(SmartFormConfigurationProperty.Title,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            List<SmartFormConfigurationData> smartFormConfigurationDataList =
smartFormConfigurationManager.GetList(criteria);

            uxSmartFormConfigurationDataListView.Visible = true;
            uxSmartFormConfigurationDataListView.DataSource =
```

```
smartFormConfigurationDataList;
            uxSmartFormConfigurationDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.SmartFormConfigurationData)
```

Updates an existing SmartFormConfigurationData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Smart Form ID
- Title
- Type
- Description

Parameters

- `smartFormConfigurationData`. The SmartFormConfigurationData object to update.

Returns

Returns the custom CmsData object added.

Remarks

Validate the Smart Form configuration item with `GetItem()` before you update the properties. Then, call `Update` with your modified SmartFormConfigurationData object.

NOTE: You cannot change all properties after the initial Add event, such as the SmartFormConfigurationData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxSmartFormIdLabel" AssociatedControlID="uxSmartFormId"
CssClass="span-3 last"
            runat="server" Text="*SmartFormId :" />
        <ektronUI:TextField ID="uxSmartFormId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-3
last"
            runat="server" Text="Title:" />
        <ektronUI:TextField ID="uxTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTypeLabel" AssociatedControlID="uxType" CssClass="span-3
last"
            runat="server" Text="Type:" />
        <asp:DropDownList ID="uxType" runat="server">
            <asp:ListItem Value="0">Content</asp:ListItem>
            <asp:ListItem Value="1">Product</asp:ListItem>
            <asp:ListItem Value="2">WebEvent</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDescriptionLabel" AssociatedControlID="uxDescription"
CssClass="span-3 last"
            runat="server" Text="Description:" />
        <asp:TextBox TextMode="MultiLine" Rows="3" ID="uxDescription" CssClass="span-4"
runat="server"
            ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
using Ektron.Cms.Framework.User;
using Microsoft.VisualBasic;
using Ektron.Cms.Framework.Content;
using System.Text;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            SmartFormConfigurationManager smartFormConfigurationManager = new
SmartFormConfigurationManager();
            long SmartformID = long.Parse(uxSmartFormId.Text);
            UserManager userManager = new UserManager();
            if (userManager.UserId > 0)
            {
                SmartFormConfigurationData smartFormConfigurationData =
smartFormConfigurationManager.GetItem(SmartformID);

                if (smartFormConfigurationData != null)
                {
                    smartFormConfigurationData.SmartformTitle = uxTitle.Text !=
string.Empty ? uxTitle.Text : smartFormConfigurationData.SmartformTitle;
                    smartFormConfigurationData.SmartformDescription = uxDescription.Text
!= string.Empty ? uxDescription.Text : smartFormConfigurationData.SmartformDescription;
                    smartFormConfigurationData.Type = (EkEnumeration.XmlConfigType)
(Convert.ToInt32(uxType.SelectedItem.Value));

                    smartFormConfigurationManager.Update(smartFormConfigurationData);
                    MessageUtilities.UpdateMessage(uxMessage,  "Smartform Configuration
with Id " + smartFormConfigurationData.Id + " has been updated.",
Message.DisplayModes.Success);
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
SmartForm ID.", Message.DisplayModes.Error);
                    return;
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Log In.",
Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
}
```

Data Classes

SmartFormConfigurationCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Content
```

Constructors

- `SmartFormConfigurationCriteria()`

  ```
  public SmartFormConfigurationCriteria()
  ```

- `SmartFormConfigurationCriteria (Ektron.Cms.Common.SmartFormConfigurationProperty, EkEnumeration.OrderByDirection)`

  ```
  public SmartFormConfigurationCriteria
  (Ektron.Cms.Common.SmartFormConfigurationProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `SmartFormConfigurationProperty` are:

  - DefaultXslt
  - Description
  - DisplatyXslt1 (DisplayXslt1)
  - DisplatyXslt2 (DisplayXslt2)
  - DisplatyXslt3 (DisplayXslt3)
  - DisplatyXslt4 (DisplayXslt4)
  - DisplatyXslt5 (DisplayXslt5)
  - EditXslt
  - FieldList
  - Id
  - LogicalPath
  - PackageDisplayXslt
  - PackageXslt
  - PhysicalPath
  - SaveXslt

- SecondType
- Title
- Type
- UserId
- XmlAdvConfig
- XmlNameSpace
- XmlSchema

SmartFormConfigurationData

Namespace

```
Ektron.Cms
```

Properties

- `DefaultXslt`. Display XSLT that being assigned from collection of display XSLTs.

  ```
  public int DefaultXslt { set; get; }
  ```

- `FieldList`. String that contains all the SmartForm Fields

  ```
  public string FieldList { set; get; }
  ```

- `LogicalPath`. SmartForm LogicalPath

  ```
  public string LogicalPath { set; get; }
  ```

- `PackageDisplayXslt`

  ```
  public string PackageDisplayXslt { set; get; }
  ```

- `PackageXslt`. Display XSLT Package (created using the Data Designer)

  ```
  public string PackageXslt { set; get; }
  ```

- `PhysicalPath`. SmartForm PhysicalPath

  ```
  public string PhysicalPath { set; get; }
  ```

- `SecondType`. public EkEnumeration.CatalogEntryType SecondType { set; get; }

  ```
  public EkEnumeration.CatalogEntryType SecondType { set; get; }
  ```

- `SmartformDescription`. Detailed description of configuration given by creator or last editor.

  ```
  public string SmartformDescription { set; get; }}
  ```

- `SmartformTitle`. SmartForm Title

  ```
  public string SmartformTitle { set; get; }
  ```

- `Type`. Set the Smartform Type(Content = 0,Product = 1,WebEvent = 2)

  ```
  public EkEnumeration.XmlConfigType Type { set; get; }
  ```

- `Validate()`

  ```
  public override ValidationResults Validate()
  ```

- `XmlSchema`. Display XSLT 1

  ```
  public string XmlSchema { set; get; }
  ```

- `Xslt1`. Display XSLT 1.
```
public string Xslt1 { set; get; }
```
- `Xslt2`. Display XSLT 2.
```
public string Xslt2 { set; get; }
```
- `Xslt3`. Display XSLT 3.
```
public string Xslt3 { set; get; }
```
- `Xslt4`. Display XSLT 4.
```
public string Xslt4 { set; get; }
```
- `Xslt5`. Display XSLT 5.
```
public string Xslt5 { set; get; }
```

TaskCommentManager

8.60 and higher

The TaskCommentManager class manages task comments. See Assigning and Managing Tasks in the Ektron Reference for information about tasks.

Namespace

```
Ektron.Cms.Framework.Settings.TaskCommentManager
```

Constructors

- `TaskCommentManager()`
- `TaskCommentManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.TaskCommentData)
```

Adds a task comment based on information in an TaskCommentData object. The method populates the message.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title

Parameters

- `TaskCommentData`. The TaskCommentData object to add.

Returns

Added TaskCommentData object.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TaskCommentManager taskCommentManager = new TaskCommentManager();
            TaskCommentData taskCommentData = new TaskCommentData()
            {
                TaskComment = uxTaskComment.Text,
                UserId = long.Parse(uxUserId.Text)
            };

             TaskCommentData retData= taskCommentManager.Add(taskCommentData);


            MessageUtilities.UpdateMessage(uxMessage, "Task added with Id " +
retData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
```

```
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a task comment from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The ID of the task comment to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taskCommentId = long.Parse(uxTaskCommentId.Text);

            TaskCommentManager taskCommentManager = new TaskCommentManager();
            TaskCommentData taskCommentData = taskCommentManager.GetItem(taskCommentId);

             if (taskCommentData != null)
             {
                 taskCommentManager.Delete(taskCommentId);
                 MessageUtilities.UpdateMessage(uxMessage, "Task comment deleted.",
Message.DisplayModes.Success);
             }
             else
             {
                 MessageUtilities.UpdateMessage(uxMessage,"Task comment does not
exists.", Message.DisplayModes.Error);
             }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
```

```
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.TaskCommentData)
```

Retrieves the properties of a specific TaskCommentData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the task comment to retrieve.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taskCommentId = long.Parse(uxTaskCommentId.Text);

            TaskCommentManager taskCommentManager = new TaskCommentManager();
            TaskCommentData taskCommentData = taskCommentManager.GetItem(taskCommentId);

            if (taskCommentData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for task comment with
ID " + taskCommentData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxTaskId.Text = "TaskId : " + taskCommentData.TaskId;
                uxTaskComment.Text = "TaskComment : " + taskCommentData.TaskComment;
                uxUserId.Text = "UserID : " + taskCommentData.UserId;
                uxCommentedDate.Text = "CreatedDate : " + taskCommentData.Commenteddate;
                uxModifiedDate.Text = "Modified Date : " +
taskCommentData.CommentUpdateddate;

            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Task comment does not
exists.", Message.DisplayModes.Error);
```

```
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.TaskCommentCriteria)
```

Retrieves lists of objects through the GetList(TaskCommentCriteria) method.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Task Category Property
- * Object Value

Parameters

- `criteria`. Criteria used to retrieve task categories.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            TaskCommentManager taskCommentManager = new TaskCommentManager();

            TaskCommentCriteria criteria = new TaskCommentCriteria();
            if (uxTaskCommentDataProperty.SelectedItem.Text == "TaskId")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaskCommentProperty.TaskId ,
CriteriaFilterOperator.Contains, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaskCommentProperty.Comments,
CriteriaFilterOperator.EqualTo, Objectvalue);
```

```
            }

            List<TaskCommentData> taskCommentDataList = taskCommentManager.GetList
(criteria);

            uxTaskDataListView.Visible = true;
            uxTaskDataListView.DataSource = taskCommentDataList;
            uxTaskDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.SmartFormConfigurationData)
```

Updates an existing task comment item in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Title

Parameters

- `TaskCommentData`. The TaskCommentData object to update.

Returns

Updated TaskCommentData object.

Remarks

Validate the task comment item with `GetItem()` before you update the properties. Then, call `Update` with your modified TaskCommentData object.

NOTE: You cannot change all properties after the initial Add event, such as the TaskCommentData.Type.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taskCommentId = long.Parse(uxTaskCommentId.Text);

            TaskCommentManager taskCommentManager = new TaskCommentManager();
            TaskCommentData taskCommentData = taskCommentManager.GetItem(taskCommentId);

           if (taskCommentData != null)
            {
                taskCommentData.TaskComment = uxTaskComment.Text;
                taskCommentManager.Update(taskCommentData);

                MessageUtilities.UpdateMessage(uxMessage, "Task comment updated",
Message.DisplayModes.Success);
            }
           else
            {
               MessageUtilities.UpdateMessage(uxMessage, "Task comment does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

TaskCategoryManager

8.50 and higher

The TaskCategoryManager class manages task categories. See Assigning and Managing Tasks in the Ektron Reference for information about tasks.

Namespace

```
Ektron.Cms.Framework.Settings.TaskCategoryManager
```

Constructors

- `TaskCategoryManager()`
- `TaskCategoryManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `TaskCategoryManagerService`. Returns an instance of the business objects task manager service.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 1602
- *GetItem* on page 1604
- *GetList* on page 1605
- *Update* on page 1608

Add

```
Add(Ektron.Cms.taskCategoryData)
```

Adds a task category based on information in a TaskCategoryData object. The method populates the message.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title

Parameters

- `taskCategoryData`. The TaskCategoryData object to add.

Returns

Added TaskCategoryData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-3
last" runat="server" Text="* Title:" />
        <ektronUI:TextField  ID="uxTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TaskCategoryManager taskCategoryManager = new TaskCategoryManager();

            TaskCategoryData taskCategoryData = new TaskCategoryData()
            {
                Title = uxTitle.Text
            };

            taskCategoryManager.Add(taskCategoryData);

            MessageUtilities.UpdateMessage(uxMessage, "Task category added with Id " +
taskCategoryData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a task category from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. The ID of the task category to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxTaskCategoryIdLabel"
AssociatedControlID="uxTaskCategoryId" CssClass="span-3 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField ID="uxTaskCategoryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taskTypeId = long.Parse(uxTaskCategoryId.Text);

            TaskCategoryManager taskCategoryManager = new TaskCategoryManager();
            TaskCategoryData taskCategoryData = taskCategoryManager.GetItem(taskTypeId);

            if (taskCategoryData != null)
            {
                taskCategoryManager.Delete(taskTypeId);
                MessageUtilities.UpdateMessage(uxMessage, "Task category deleted.",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Task category does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.TaskCategoryData)
```

Retrieves the properties of a specific TaskCategoryData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the task category to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaskCategoryIdLabel"
AssociatedControlID="uxTaskCategoryId" CssClass="span-3 last" runat="server" Text="* Id
:" />
        <ektronUI:TextField ID="uxTaskCategoryId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxActive" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taskId = long.Parse(uxTaskCategoryId.Text);

            TaskCategoryManager taskCategoryManager = new TaskCategoryManager();
            TaskCategoryData taskCategoryData = taskCategoryManager.GetItem(taskId);

            if (taskCategoryData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for task category
with ID " + taskId.ToString() + " are below", Message.DisplayModes.Success);

                uxTitle.Text = "Title : " + taskCategoryData.Title ;
                uxActive.Text = "Active : " + taskCategoryData.IsActive;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Task category does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.TaskCategoryCriteria)
```

Retrieves lists of objects through the GetList(TaskCategoryCriteria) method.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Task Category Property
- * Object Value

Parameters

- `criteria`. Criteria used to retrieve task categories.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaskCategoryPropertyLabel"
AssociatedControlID="uxTaskCategoryProperty" CssClass="span-5 last" runat="server"
Text="* TaskCategoryProperty:" />
        <asp:DropDownList ID="uxTaskCategoryProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Title</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-5 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxTaskCategoryDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Title
                    </th>
                    <th>
                        Active
                    </th>
```

```
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Title")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsActive")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            TaskCategoryManager taskCategoryManager = new TaskCategoryManager();

            TaskCategoryCriteria criteria = new TaskCategoryCriteria();
            if (uxTaskCategoryProperty.SelectedItem.Text == "Id")
            {
                Objectvalue = long.Parse(uxObjectValue.Text);
                criteria.AddFilter(TaskCategoryProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaskCategoryProperty.Title,
CriteriaFilterOperator.Contains, Objectvalue);
```

```
            }

            List<TaskCategoryData> taskCategoryDataList = taskCategoryManager.GetList
(criteria);

            uxTaskCategoryDataListView.Visible = true;
            uxTaskCategoryDataListView.DataSource = taskCategoryDataList;
            uxTaskCategoryDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.SmartFormConfigurationData)
```

Updates an existing TaskCategoryData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Title

Parameters

- `taskCategoryData`. The TaskCategoryData object to update.

Returns

Updated TaskCategoryData object.

Remarks

Validate the task category item with `GetItem()` before you update the properties. Then, call `Update` with your modified TaskCategoryData object.

NOTE: You cannot change all properties after the initial Add event, such as the TaskCategoryData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaskCategoryIdLabel"
AssociatedControlID="uxTaskCategoryId" CssClass="span-3 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField  ID="uxTaskCategoryId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-3
last" runat="server" Text="* Title:" />
        <ektronUI:TextField  ID="uxTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taskCategoryId = long.Parse(uxTaskCategoryId.Text);

            TaskCategoryManager taskCategoryManager = new TaskCategoryManager();
            TaskCategoryData taskCategoryData = taskCategoryManager.GetItem
(taskCategoryId);

            if (taskCategoryData != null)
            {
                taskCategoryData.Title  = (uxTitle.Text != string.Empty ? uxTitle.Text :
taskCategoryData.Title);
                taskCategoryManager.Update(taskCategoryData);

                MessageUtilities.UpdateMessage(uxMessage, "Task category updated",
```

```
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Task category does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

TaskCategoryCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms
```

Constructors

- `TaskCategoryCriteria()`

  ```
  public TaskCategoryCriteria()
  ```

- `TaskCategoryCriteria(Ektron.Cms.Common.TaskCategoryProperty, EkEnumeration.OrderByDirection)`

  ```
  public TaskCategoryCriteria(Ektron.Cms.Common.TaskCategoryProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `TaskCategoryProperty` are:

  - Active
  - Id
  - Title

TaskCategoryData

Namespace

```
Ektron.Cms
```

Properties

- `Active`

  ```
  public int Active
  ```

- `ID`

```
public long ID { set; get; }
```

- `IsActive`. Gets or sets the Active flag.

```
public bool IsActive { set; get; }
```

- `Title`. Gets or sets the title of the Task Category.

```
public string Title { set; get; }
```

- `TaskCategoryData()`

```
public TaskCategoryData()
```

- `Types`

```
public Ektron.Cms.TaskTypeData[] Types { set; get; }
```

TaskManager

8.50 and higher

The TaskManager class manages tasks. See Assigning and Managing Tasks in the Ektron Reference for information about tasks.

Namespace

```
Ektron.Cms.Framework.Settings.TaskManager
```

Constructors

- `TaskManager()`
- `TaskManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `TaskManagerService`. Returns an instance of the business objects task manager service.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 1616
- *GetItem* on page 1618
- *GetList* on page 1620
- *Update* on page 1622

Add

```
Add(Ektron.Cms.TaskData)
```

Adds a task based on information in a TaskData object. The method populates the message.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Title
- * Priority
- * State
- * User ID
- * Content ID
- * Start Date Time
- * End Date Time
- Description

Parameters

- `taskData`. The TaskData object to add.

Returns

The added TaskData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-3
last" runat="server" Text="* Title:" />
        <ektronUI:TextField  ID="uxTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPriorityLabel" AssociatedControlID="uxPriority"
CssClass="span-3 last" runat="server" Text="* Priority:" />
        <asp:DropDownList ID="uxPriority" runat="server">
            <asp:ListItem Value="1">Low</asp:ListItem>
            <asp:ListItem Value="2">Normal</asp:ListItem>
            <asp:ListItem Value="3">High</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStateLabel" AssociatedControlID="uxState" CssClass="span-3
last" runat="server" Text="* State:" />
        <asp:DropDownList ID="uxState" runat="server">
            <asp:ListItem Value="1">NotStarted</asp:ListItem>
```

```
            <asp:ListItem Value="2">Active</asp:ListItem>
            <asp:ListItem Value="3">AwaitingData</asp:ListItem>
            <asp:ListItem Value="4">OnHold</asp:ListItem>
            <asp:ListItem Value="5">Pending</asp:ListItem>
            <asp:ListItem Value="6">Reopened</asp:ListItem>
            <asp:ListItem Value="7">Completed</asp:ListItem>
            <asp:ListItem Value="8">Archived</asp:ListItem>
            <asp:ListItem Value="9">Deleted</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last" runat="server" Text="* User Id:" />
        <ektronUI:TextField  ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text ="10005" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField  ID="uxContentId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-3 last"
            runat="server" Text="*Start DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEndDateLabel" AssociatedControlID="uxEndDate"
CssClass="span-3 last"
            runat="server" Text="*End DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxEndDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxEndTime" runat="server">
```

```
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
            <asp:ListItem>12</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEndPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxTaskDescriptionLabel"
AssociatedControlID="uxTaskDescription" CssClass="span-3 last" runat="server"
Text="Description :" />
        <asp:TextBox TextMode="MultiLine" Rows="3" ID="uxTaskDescription"
CssClass="span-4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TaskManager taskManager = new TaskManager();

            TaskData taskData = new TaskData()
```

```
            {
                TaskTitle = uxTitle.Text,
                LanguageID = taskManager.RequestInformation.ContentLanguage,
                AssignedToUserID = long.Parse(uxUserId.Text),
                AssignorUserId = taskManager.RequestInformation.UserId,
                Priority = (EkEnumeration.TaskPriority)int.Parse
(uxPriority.SelectedItem.Value),
                ContentID = long.Parse(uxContentId.Text),
                ContentLanguage = taskManager.RequestInformation.ContentLanguage,
                State = int.Parse(uxState.SelectedItem.Value).ToString(),
                StartDate = DateTime.Parse(uxStartDate.Text + " " +
uxStartTime.SelectedItem.Text + ":00:00 " + uxStartPeriod.SelectedItem.Text).ToString(),
                DueDate = DateTime.Parse(uxEndDate.Text + " " +
uxEndTime.SelectedItem.Text + ":00:00 " + uxEndPeriod.SelectedItem.Text).ToString() ,
                Description  = uxTaskDescription.Text
            };

            taskManager.Add(taskData);

            MessageUtilities.UpdateMessage(uxMessage, "Task added with Id " +
taskData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a task from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the task to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaskIdLabel" AssociatedControlID="uxTaskId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTaskId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taskId = long.Parse(uxTaskId.Text);

            TaskManager taskManager = new TaskManager();
            TaskData taskData = taskManager.GetItem(taskId);

             if (taskData != null)
             {
                 taskManager.Delete(taskId);
                 MessageUtilities.UpdateMessage(uxMessage, "Task deleted.",
Message.DisplayModes.Success);
             }
             else
             {
                 MessageUtilities.UpdateMessage(uxMessage,"Task does not exists.",
Message.DisplayModes.Error);
             }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
```

```
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.TaskData)
```

Retrieves the properties of a specific TaskData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the task to retrieve.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaskIdLabel" AssociatedControlID="uxTaskId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxTaskId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTitle" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAssignedToUserID" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxStatus" runat="server"></asp:Literal>
```

```
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxPriority" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxStartDate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDueDate" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taskId = long.Parse(uxTaskId.Text);

            TaskManager taskManager = new TaskManager();
            TaskData taskData = taskManager.GetItem(taskId);

            if (taskData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage,"Details for task with ID " +
taskData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxTitle.Text = "Title : " + taskData.TaskTitle;
                uxAssignedToUserID.Text = "Assigned To User ID : " +
taskData.AssignedToUserID;
                uxStatus.Text = "State : " + taskData.State;
                uxPriority.Text = "Priority : " +
(EkEnumeration.TaskPriority)taskData.Priority;
                uxStartDate.Text = "Start Date : " + taskData.StartDate;
                uxDueDate.Text = "Due Date : " + taskData.DueDate ;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Task does not exists.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

```
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.TaskCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Task Data Property
- * Object Value

Parameters

- `criteria`. TaskCriteria used to retrieve tasks.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaskDataPropertyLabel"
AssociatedControlID="uxTaskDataProperty" CssClass="span-6 last" runat="server"
Text="TaskDataProperty:" />
        <asp:DropDownList ID="uxTaskDataProperty" runat="server">
            <asp:ListItem>Title</asp:ListItem>
            <asp:ListItem>Priority(2,1,0)</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxTaskDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Task Title
                    </th>
                    <th>
                        Priority
                    </th>
                    <th>
                        Content ID
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("TaskID")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TaskTitle")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Priority")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ContentID")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            TaskManager taskManager = new TaskManager();

            TaskCriteria criteria = new TaskCriteria();
            if (uxTaskDataProperty.SelectedItem.Text == "Title")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaskProperty.Title, CriteriaFilterOperator.Contains,
Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(TaskProperty.Priority,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            List<TaskData> taskDataList = taskManager.GetList(criteria);

            uxTaskDataListView.Visible = true;
            uxTaskDataListView.DataSource = taskDataList;
            uxTaskDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.TaskData)
```

Updates an existing TaskData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Title
- * Priority
- * State
- * User ID
- * Content ID
- * Start Date Time
- * End Date Time
- Description

Parameters

- `taskData`. The TaskData object to add.

Returns

The added TaskData object.

Remarks

Validate the task item with `GetItem()` before you update the properties. Then, call `Update` with your modified TaskData object.

NOTE: You cannot change all properties after the initial Add event, such as the TaskData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaskIdLabel" AssociatedControlID="uxTaskId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField  ID="uxTaskId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTitleLabel" AssociatedControlID="uxTitle" CssClass="span-3
last" runat="server" Text="* Title:" />
        <ektronUI:TextField  ID="uxTitle" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPriorityLabel" AssociatedControlID="uxPriority"
CssClass="span-3 last" runat="server" Text="* Priority:" />
        <asp:DropDownList ID="uxPriority" runat="server">
            <asp:ListItem Value="1">Low</asp:ListItem>
            <asp:ListItem Value="2">Normal</asp:ListItem>
            <asp:ListItem Value="3">High</asp:ListItem>
        </asp:DropDownList>
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStateLabel" AssociatedControlID="uxState" CssClass="span-3
last" runat="server" Text="* State:" />
        <asp:DropDownList ID="uxState" runat="server">
            <asp:ListItem Value="1">NotStarted</asp:ListItem>
            <asp:ListItem Value="2">Active</asp:ListItem>
            <asp:ListItem Value="3">AwaitingData</asp:ListItem>
            <asp:ListItem Value="4">OnHold</asp:ListItem>
            <asp:ListItem Value="5">Pending</asp:ListItem>
            <asp:ListItem Value="6">Reopened</asp:ListItem>
            <asp:ListItem Value="7">Completed</asp:ListItem>
            <asp:ListItem Value="8">Archived</asp:ListItem>
            <asp:ListItem Value="9">Deleted</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last" runat="server" Text="* User Id:" />
        <ektronUI:TextField  ID="uxUserId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text ="10005" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField  ID="uxContentId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxStartDateLabel" AssociatedControlID="uxStartDate"
CssClass="span-3 last"
            runat="server" Text="*Start DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxStartDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxStartTime" runat="server">
            <asp:ListItem>12</asp:ListItem>
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxStartPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEndDateLabel" AssociatedControlID="uxEndDate"
```

```
CssClass="span-3 last"
            runat="server" Text="*End DateTime :" />
        <ektronUI:DatePicker Rows="3" ID="uxEndDate" CssClass="span-4" runat="server"
Text="MM/DD/YYYY"
            ValidationGroup="RegisterValidationGroup" />
        <asp:DropDownList ID="uxEndTime" runat="server">
            <asp:ListItem>1</asp:ListItem>
            <asp:ListItem>2</asp:ListItem>
            <asp:ListItem>3</asp:ListItem>
            <asp:ListItem>4</asp:ListItem>
            <asp:ListItem>5</asp:ListItem>
            <asp:ListItem>6</asp:ListItem>
            <asp:ListItem>7</asp:ListItem>
            <asp:ListItem>8</asp:ListItem>
            <asp:ListItem>9</asp:ListItem>
            <asp:ListItem>10</asp:ListItem>
            <asp:ListItem>11</asp:ListItem>
            <asp:ListItem>12</asp:ListItem>
        </asp:DropDownList>
        <asp:DropDownList ID="uxEndPeriod" runat="server">
            <asp:ListItem>AM</asp:ListItem>
            <asp:ListItem>PM</asp:ListItem>
        </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxTaskDescriptionLabel"
AssociatedControlID="uxTaskDescription" CssClass="span-3 last" runat="server"
Text="Description :" />
        <asp:TextBox TextMode="MultiLine" Rows="3" ID="uxTaskDescription"
CssClass="span-4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taskId = long.Parse(uxTaskId.Text);

            TaskManager taskManager = new TaskManager();
            TaskData taskData = taskManager.GetItem(taskId);

           if (taskData != null)
            {
                taskData.TaskTitle = (uxTitle.Text != string.Empty ? uxTitle.Text
:taskData.TaskTitle);
                taskData.AssignedToUserID =( uxUserId.Text != string.Empty ? long.Parse
(uxUserId.Text): taskData.AssignedToUserID);
                taskData.AssignorUserId  = taskManager.RequestInformation.UserId;
                taskData.Priority = (EkEnumeration.TaskPriority)int.Parse
(uxPriority.SelectedItem.Value);
                taskData.ContentID = (uxContentId.Text != string.Empty ?long.Parse
(uxContentId.Text): taskData.ContentID);
                taskData.ContentLanguage =
taskManager.RequestInformation.ContentLanguage;
                taskData.State = int.Parse(uxState.SelectedItem.Value).ToString();
                taskData.StartDate = (uxStartDate.Text != string.Empty ? DateTime.Parse
(uxStartDate.Text + " " + uxStartTime.SelectedItem.Text + ":00:00 " +
uxStartPeriod.SelectedItem.Text).ToString() :taskData.StartDate);
                taskData.DueDate = (uxEndDate.Text != string.Empty ? DateTime.Parse
(uxEndDate.Text + " " + uxEndTime.SelectedItem.Text + ":00:00 " +
uxEndPeriod.SelectedItem.Text).ToString() :  taskData.DueDate);
                taskData.Description = uxTaskDescription.Text != string.Empty ?
uxTaskDescription.Text : taskData.Description;

                taskManager.Update(taskData);

                MessageUtilities.UpdateMessage(uxMessage, "Task updated",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Task does not exists.",
Message.DisplayModes.Error);
            }
             uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

TaskCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms
```

Constructors

- `TaskCriteria()`

  ```
  public TaskCriteria()
  ```

- `TaskCriteria(Ektron.Cms.Common.TaskProperty, EkEnumeration.OrderByDirection)`

  ```
  public TaskCriteria(Ektron.Cms.Common.TaskProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `TaskProperty` are:

  - AssignedToGroupId
  - AssignedToUserId
  - ContentId
  - Description
  - DueDate
  - Id
  - Priority
  - Title

TaskData

Namespace

```
Ektron.Cms
```

Properties

- `AssignedByUser`

  ```
  public string AssignedByUser { set; get; }
  ```

- `AssignedByUserId`

  ```
  public string AssignedByUserID { set; get; }
  ```

- `AssignedToUser`. The username of the user who the task is assigned to. This value is populated when retrieving TaskData. While adding a new Task, use the AssignedToUserID property.

  ```
  public string AssignedToUser { set; get; }
  ```

- `AssignedToUserGroup`

  ```
  public string AssignedToUserGroup { set; get; }
  ```

- `AssignedToUserID`. The ID of the user to assign the task to. This is a required property to successfully assign a task to a user.

```
public long AssignedToUserID { set; get; }
```

- `Assignor`. The username of the user assigning the task.

```
public string Assignor { set; get; }
```

- `AssignorUserId`. The ID of the user assigning the task.

```
public long AssignorUserId { set; get; }
```

- `AssignToUserGroupID`

```
public long AssignToUserGroupID { set; get; }
```

- `CommentDisplayName`

```
public string CommentDisplayName { set; get; }
```

- `CommentEmail`

```
public string CommentEmail { set; get; }
```

- `CommentURI`

```
public string CommentURI { set; get; }
```

- `ContentID`

```
public long ContentID { set; get; }
```

- `ContentLanguage`

```
public int ContentLanguage { set; get; }
```

- `ContentTitle`

```
public string ContentTitle { set; get; }
```

- `ContentType`

```
public EkEnumeration.CMSContentType ContentType { set; get; }
```

- `CreatedByUser`

```
public string CreatedByUser { set; get; }
```

- `CreatedByUserID`

```
public long CreatedByUserID { set; get; }
```

- `DateCreated`

```
public string DateCreated { set; get; }
```

- `DateModified`

```
public string DateModified { set; get; }
```

- `Description`

```
public string Description { set; get; }
```

- `DisplayDateCreated`

```
public string DisplayDateCreated { set; get; }
```

- `DisplayDateModified`

```
public string DisplayDateModified { set; get; }
```

- `DueDate`

```
public string DueDate { set; get; }
```

- `FolderId`

```
public long FolderId { set; get; }
```

- Id

```
public override long Id { set; get; }
```

- ImpersonateUser

```
public bool ImpersonateUser { set; get; }
```

- Language

```
public string Language { set; get; }
```

- LanguageID

```
public int LanguageID { set; get; }
```

- LastComment

```
public string LastComment { set; get; }
```

- LastCommentUserID

```
public long LastCommentUserID { set; get; }
```

- ParentID

```
public long ParentID { set; get; }
```

- Priority

```
public EkEnumeration.TaskPriority Priority { set; get; }
```

- StartDate

```
public string StartDate { set; get; }
```

- State. One of the following values from Common.EkEnumeration.TaskState:
  - Prototype = "0"
  - NotStarted = "1"
  - Active = "2"
  - AwaitingData = "3"
  - OnHold = "4"
  - Pending = "5"
  - Reopened = "6"
  - Completed = "7"
  - Archived = "8"
  - Deleted = "9"

```
public string State { set; get; }
```

- Status. Obsolete: Use State

```
public int Status { set; get; }
```

- TaskID

```
public long TaskID { set; get; }
```

- TaskTitle

```
public string TaskTitle { set; get; }
```

- TaskTypeID

```
public long TaskTypeID { set; get; }
```

- `UserInfo`

```
public Ektron.Cms.UserData UserInfo { set; get; }
```

- `Validate()`

```
public override ValidationResults Validate()
```

TaxonomyCustomPropertyManager

8.50 and higher

The TaxonomyCustomPropertyManager class manages user-specified taxonomy properties.

Namespace

```
Ektron.Cms.Framework.Settings.TaxonomyCustomPropertyManager
```

Constructors

- `TaxonomyCustomPropertyManager()`
- `TaxonomyCustomPropertyManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.CustomPropertyData)
```

Adds a taxonomy custom property.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Name
- * Type
- Is Enabled
- Is Editable

Parameters

- `data`. CustomPropertyData object with the details to add.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-3
last" runat="server" Text="* Name:" />
        <ektronUI:TextField  ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTypeLabel" AssociatedControlID="uxType" CssClass="span-3
last" runat="server" Text="* Type:" />
        <asp:DropDownList ID="uxType" runat="server">
            <asp:ListItem Value="0">String</asp:ListItem>
            <asp:ListItem Value="2">Numeric</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-3 last" runat="server" Text="Is Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEnabled" CssClass="span-
6" runat="server"  ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEditableLabel" AssociatedControlID="uxIsEditable"
CssClass="span-3 last" runat="server" Text="Is Editable:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEditable"
CssClass="span-6" runat="server"  ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            TaxonomyCustomPropertyManager taxonomyCustomPropertyManager = new
TaxonomyCustomPropertyManager();

            CustomPropertyData customPropertyData = new CustomPropertyData()
            {
                PropertyName = uxName.Text,
                CmsObjectType = EkEnumeration.CustomPropertyObjectType.TaxonomyNode,
                LanguageId =
taxonomyCustomPropertyManager.RequestInformation.ContentLanguage,
                PropertyDataType =
(EkEnumeration.CustomPropertyItemDataType)Convert.ToInt32(uxType.SelectedItem.Value),
                PropertyStyleType = EkEnumeration.CustomPropertyStyleType.SingleSelect,
                IsEditable = uxIsEditable.Checked,
                IsEnabled = uxIsEnabled.Checked,
            };

            taxonomyCustomPropertyManager.Add(customPropertyData);

            MessageUtilities.UpdateMessage(uxMessage,  "Taxonomy custom property added
with Id " + customPropertyData.PropertyId, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a taxonomy custom property from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Language ID

Parameters

- `propertyId`. ID of the CustomPropertyData object to delete.
- `languageId`. Language Id of the CustomProperty to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyCustomUserPropertyIdLabel"
AssociatedControlID="uxTaxonomyCustomUserPropertyId" CssClass="span-4 last"
runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxTaxonomyCustomUserPropertyId" CssClass="span-6"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageIdLabel" AssociatedControlID="uxLanguageId"
CssClass="span-4 last" runat="server" Text="* Language Id:" />
        <ektronUI:TextField ID="uxLanguageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long taxonomyCustomUserPropertyId = long.Parse
(uxTaxonomyCustomUserPropertyId.Text);
            int languageId = int.Parse(uxLanguageId.Text);

            TaxonomyCustomPropertyManager taxonomyCustomPropertyManager = new
TaxonomyCustomPropertyManager();
            CustomPropertyData customPropertyData =
taxonomyCustomPropertyManager.GetItem(taxonomyCustomUserPropertyId, languageId);

            if (customPropertyData != null && customPropertyData.PropertyId > 0)
             {
                taxonomyCustomPropertyManager.Delete(taxonomyCustomUserPropertyId,
languageId);
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy custom property
deleted.", Message.DisplayModes.Success);
             }
             else
             {
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy custom property
does not exists.", Message.DisplayModes.Error);
             }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.CustomPropertyData)
```

Retrieves the properties of a specific CustomPropertyData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Language ID

Parameters

- `propertyId`. ID of theCustomPropertyData object.
- `languageId`. Language ID of the CustomPropertyData object to get.

Returns

CustomProperty details in a CustomPropertyData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxPropertyIdLabel" AssociatedControlID="uxPropertyId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxPropertyId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageIdLabel" AssociatedControlID="uxLanguageId"
CssClass="span-3 last" runat="server" Text="* Language Id :" />
        <ektronUI:TextField ID="uxLanguageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDisplayValueType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsEditable" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
```

```
        <asp:Literal ID="uxIsEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLanguageIdVal" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long propertyId = long.Parse(uxPropertyId.Text);
            int languageId = int.Parse(uxLanguageId.Text);
            TaxonomyCustomPropertyManager taxonomyCustomPropertyManager = new
TaxonomyCustomPropertyManager();
            CustomPropertyData customPropertyData =
taxonomyCustomPropertyManager.GetItem(propertyId, languageId);

            if (customPropertyData != null && customPropertyData.PropertyId > 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for taxonomy custom
property with ID " + customPropertyData.PropertyId.ToString() + " are below",
Message.DisplayModes.Success);

                uxName.Text = "Name : " + customPropertyData.PropertyName;
                uxDisplayValueType.Text = "Property Data Type : " +
customPropertyData.PropertyDataType;
                uxIsEditable.Text = "Is Editable : " + customPropertyData.IsEditable;
                uxIsEnabled.Text = "Is Enabled : " + customPropertyData.IsEnabled;
                uxLanguageIdVal.Text = "Language Id : " + customPropertyData.LanguageId;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy custom property does
not exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

GetList (paging info)

```
GetList(System.Int32,Ektron.Cms.PagingInfo)
```

Retrieves a list of CustomPropertyData objects with their details.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Language ID

Parameters

- `languageId`. Language ID of the CustomProperties to get.
- `paging`. Paging details.

Returns

List of CustomPropertyData objects.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageIdLabel" AssociatedControlID="uxLanguageId"
CssClass="span-3 last" runat="server" Text="* Language Id:" />
        <ektronUI:TextField ID="uxLanguageId" CssClass="span-3" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxCustomPropertyDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
```

```
                        Property Name
                    </th>
                    <th>
                        Language Id
                    </th>
                    <th>
                        Property Id
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("PropertyName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("LanguageId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("PropertyId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            int languageId = int.Parse(uxLanguageId.Text);

            PagingInfo pagingInfo = new PagingInfo();
            pagingInfo.RecordsPerPage = 100;

            TaxonomyCustomPropertyManager taxonomyCustomPropertyManager = new
TaxonomyCustomPropertyManager();
```

```
            List<CustomPropertyData> customPropertyData =
taxonomyCustomPropertyManager.GetList(languageId, pagingInfo);

            uxCustomPropertyDataListView.Visible = true;
            uxCustomPropertyDataListView.DataSource = customPropertyData;
            uxCustomPropertyDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (properties)

```
GetList(Ektron.Cms.Common.EkEnumeration.CustomPropertyObjectType,
System.Int32,Ektron.Cms.PagingInfo)
```

Retrieves a list of CustomPropertyData objects with their details.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Custom Property
- * Language ID
- * Records Per Page

Parameters

- `cmsObjectType`. Type of the CustomProperties to get.
- `languageId`. Language ID of the CustomProperties to get.
- `paging`. Paging details.

Returns

List of CustomPropertyData objects.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxPropertyIdLabel" AssociatedControlID="uxPropertyId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxPropertyId" CssClass="span-6" runat="server"
```

```
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageIdLabel" AssociatedControlID="uxLanguageId"
CssClass="span-3 last" runat="server" Text="* Language Id :" />
        <ektronUI:TextField ID="uxLanguageId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDisplayValueType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsEditable" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLanguageIdVal" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            EkEnumeration.CustomPropertyObjectType objectValue =
EkEnumeration.CustomPropertyObjectType.TaxonomyNode;
```

```
            int languageId = int.Parse(uxLanguageId.Text);

            PagingInfo pagingInfo = new PagingInfo();
            pagingInfo.RecordsPerPage = int.Parse(uxRecordsPerPage.Text);

            TaxonomyCustomPropertyManager taxonomyCustomPropertyManager = new
TaxonomyCustomPropertyManager();
            List<CustomPropertyData> customPropertyData =
taxonomyCustomPropertyManager.GetList(objectValue, languageId, pagingInfo);

            uxCustomPropertyDataListView.Visible = true;
            uxCustomPropertyDataListView.DataSource = customPropertyData;
            uxCustomPropertyDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetListNonTranslated

```
GetListNonTranslated(System.Int64)
```

Retrieves a list of non-translated LanguageData objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Property ID

Parameters

- `propertyId`. ID of the CustomPropertyData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxPropertyIdLabel" AssociatedControlID="uxPropertyId"
CssClass="span-3 last" runat="server" Text="* Property Id:" />
        <ektronUI:TextField ID="uxPropertyId" CssClass="span-3" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxLanguageDataDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Name
                    </th>
                    <th>
                        SiteEnabled
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("SiteEnabled")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            int propertyId = int.Parse(uxPropertyId.Text);

            TaxonomyCustomPropertyManager taxonomyCustomPropertyManager = new
TaxonomyCustomPropertyManager();
            List<LanguageData> languageData =
taxonomyCustomPropertyManager.GetListNonTranslated(propertyId);

            uxLanguageDataDataListView.Visible = true;
            uxLanguageDataDataListView.DataSource = languageData;
            uxLanguageDataDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetListTranslated

```
GetListTranslated(System.Int64)
```

Retrieves a list of translated LanguageData objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Property ID

Parameters

- `propertyId`. ID of the CustomPropertyData object.

Returns

List of LanguageData objects.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxPropertyIdLabel" AssociatedControlID="uxPropertyId"
CssClass="span-3 last" runat="server" Text="* Property Id:" />
        <ektronUI:TextField ID="uxPropertyId" CssClass="span-3" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxLanguageDataDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Name
                    </th>
                    <th>
                        SiteEnabled
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("SiteEnabled")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            int propertyId = int.Parse(uxPropertyId.Text);

            TaxonomyCustomPropertyManager taxonomyCustomPropertyManager = new
TaxonomyCustomPropertyManager();
            List<LanguageData> languageData =
taxonomyCustomPropertyManager.GetListTranslated(propertyId);

            uxLanguageDataDataListView.Visible = true;
            uxLanguageDataDataListView.DataSource = languageData;
            uxLanguageDataDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Common.CustomPropertyData)
```

Updates an existing taxonomy CustomPropertyData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Property ID
- * Language ID
- Name
- * Is Enabled
- * Is Editable

Parameters

- `data`. CustomPropertyData object with the details to update.

Returns

Returns the custom CmsData object updated.

Remarks

Validate the taxonomy custom property item with `GetItem()` before you update the properties. Then, call `Update` with your modified CustomPropertyData object.

NOTE: You cannot change all properties after the initial Add event, such as the CustomPropertyData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxPropertyIdLabel" AssociatedControlID="uxName"
CssClass="span-3 last" runat="server" Text="* Property Id:" />
        <ektronUI:TextField  ID="uxPropertyId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageIdLabel" AssociatedControlID="uxLanguageId"
CssClass="span-3 last" runat="server" Text="* LanguageId:" />
        <ektronUI:TextField  ID="uxLanguageId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNameLabel" AssociatedControlID="uxName" CssClass="span-3
last" runat="server" Text=" Name:" />
        <ektronUI:TextField  ID="uxName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-3 last" runat="server" Text="* Is Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEnabled" CssClass="span-
6" runat="server"  ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEditableLabel" AssociatedControlID="uxIsEditable"
CssClass="span-3 last" runat="server" Text="* Is Editable:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEditable"
CssClass="span-6" runat="server"  ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long propertyId = long.Parse(uxPropertyId.Text);
            int languageId = int.Parse(uxLanguageId.Text);

            TaxonomyCustomPropertyManager taxonomyCustomPropertyManager = new
TaxonomyCustomPropertyManager();
            CustomPropertyData customPropertyData =
taxonomyCustomPropertyManager.GetItem(propertyId, languageId);

            if (customPropertyData != null)
            {
                customPropertyData.PropertyName =  (uxName.Text != string.Empty ?
uxName.Text : customPropertyData.PropertyName);
                customPropertyData.IsEditable = uxIsEditable.Checked;
                customPropertyData.IsEnabled = uxIsEnabled.Checked;

                taxonomyCustomPropertyManager.Update(customPropertyData);
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy custom property
updated. ", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Taxonomy custom property does
not exists. ", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

CustomPropertyData

Namespace

```
Ektron.Cms.Common
```

Properties

- AddItem(object, bool)

```
public void AddItem(object value, bool isDefault)
```

- CmsObjectType

```
public EkEnumeration.CustomPropertyObjectType
  CmsObjectType { set; get; }
```

- CustomPropertyData()

```
public CustomPropertyData()
```

- CustomPropertyData(string, EkEnumeration.CustomPropertyItemDataType, EkEnumeration.CustomPropertyObjectType, int)

```
public CustomPropertyData(string propertyName,
  EkEnumeration.CustomPropertyItemDataType propertyDataType,
EkEnumeration.CustomPropertyObjectType cmsObjectType, int languageId)
```

- IsEditable

```
public bool IsEditable { set; get; }
```

- IsEnabled

```
public bool IsEnabled { set; get; }
```

- Items

```
public System.Collections.Generic.List<CustomPropertyItemData>
  Items { set; get; }
```

- LanguageId

```
public int LanguageId { set; get; }
```

- PropertyDataType

```
public EkEnumeration.CustomPropertyItemDataType
  PropertyDataType { set; get; }
```

- PropertyId

```
public long PropertyId { set; get; }
```

- PropertyName

```
public string PropertyName { set; get; }
```

- PropertyStyleType

```
public EkEnumeration.CustomPropertyStyleType
  PropertyStyleType { set; get; }
```

UrlAliasing

8.50 and higher

The UrlAliasing manager category is a sub-category of Settings and manages URL aliases with the following classes:

- *AliasManager* below. Manages aliases.
- *AliasRuleManager* on page 1673. Manages alias rules.
- *AliasSettingManager* on page 1704. Manages alias settings.
- *AutoAliasManager* on page 1736. Manages automatic aliases, which lets you assign an alias to several content items at once.
- *CommonAliasManager* on page 1755. Gets alias and target data that is common to any alias.
- *CommunityAliasManager* on page 1759. Manages aliases that apply to communities.
- *ManualAliasManager* on page 1776. Manages manual aliases.
- *RedirectManager* on page 1792. Manages URL redirect addresses.
- *RegExAliasManager* on page 1812. Manages regular expressions that replace URLs that cannot be read.

AliasManager

8.61 and higher

The AliasManager class manages aliases. See Creating User-Friendly URLs with Aliasing in the Ektron Reference for information about aliases.

Namespace

```
Ektron.Cms.Framework.Settings.UrlAliasing.AliasManager
```

Constructors

- `AliasManager()`
- `AliasManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 1654
- `DeleteAll()`. Deletes all aliases.
- *GetItem* on page 1658
- *GetList* on page 1661
- *GetTarget* on page 1663
- *IsValidEktronAlias* on page 1666
- *Update* on page 1667

Add

```
Add(Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasData)
```

Adds a message based on information in an URL AliasData object.

The method populates the message.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Name
- * Extension
- * Content ID
- * Language ID

Parameters

- `data`. AliasData data.

Returns

Returns the AliasData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasNameLabel" AssociatedControlID="uxAliasName"
CssClass="span-3 last" runat="server" Text="* Alias Name:" />
        <ektronUI:TextField  ID="uxAliasName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionLabel" AssociatedControlID="uxExtension"
```

```
CssClass="span-3 last" runat="server" Text="* Extension:" />
        <asp:DropDownList ID="uxExtension" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem>.aspx</asp:ListItem>
            <asp:ListItem>/</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField  ID="uxContentId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageListLabel" AssociatedControlID="uxLanguageList"
CssClass="span-3 last" runat="server" Text="* Language Id:" />
        <asp:DropDownList ID="uxLanguageList" runat="server">
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQueryStringActionLabel"
AssociatedControlID="uxQueryStringAction" CssClass="span-3 last" runat="server" Text="
QueryString Action:" />
        <asp:DropDownList ID="uxQueryStringAction" runat="server">

            <asp:ListItem>None</asp:ListItem>
            <asp:ListItem>Append</asp:ListItem>
            <asp:ListItem>Replace</asp:ListItem>
            <asp:ListItem>Resolve</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQueryStringLabel" AssociatedControlID="uxQueryString"
CssClass="span-3 last" runat="server" Text=" Query String:" />
        <ektronUI:TextField  ID="uxQueryString" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-3 last" runat="server" Text=" Is Enabled:" />
        <asp:checkbox ID="uxIsEnabled" runat="server" Checked="true" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsDefaultLabel" AssociatedControlID="uxIsDefault"
CssClass="span-3 last" runat="server" Text=" Is Default:" />
        <asp:checkbox ID="uxIsDefault" runat="server" Checked="true" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasManager aliasManager = new AliasManager();
            string aliasName = uxAliasName.Text;
            long contentId = 0;
            long.TryParse(uxContentId.Text, out contentId);
            string queryStringAction = uxQueryStringAction.SelectedItem.Text;
            if (aliasName != "" && uxExtension.SelectedItem.Text != "Select" &&
contentId > 0)
            {
                AliasData aliasData = new AliasData();
                aliasData.Alias = aliasName + uxExtension.SelectedItem.Text;
                aliasData.Type = Ektron.Cms.Common.EkEnumeration.AliasRuleType.Manual;

                Ektron.Cms.Framework.Content.ContentManager contentManager = new
Ektron.Cms.Framework.Content.ContentManager();
                ContentData contentData = contentManager.GetItem(long.Parse
(uxContentId.Text));
                if (contentData != null && contentData.Id > 0)
                    aliasData.TargetId = contentData.Id;
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
ContentId.", Message.DisplayModes.Error);
                    uxPageMultiView.SetActiveView(uxViewMessage);
                    return;
                }

                aliasData.LanguageId = Convert.ToInt32
(uxLanguageList.SelectedItem.Value);
                if (queryStringAction == "Append")
                    aliasData.QueryStringAction =
Ektron.Cms.Common.EkEnumeration.QueryStringActionType.Append;
                else if(queryStringAction == "Replace")
                    aliasData.QueryStringAction =
Ektron.Cms.Common.EkEnumeration.QueryStringActionType.Replace;
                else if(queryStringAction == "Resolve")
                    aliasData.QueryStringAction =
Ektron.Cms.Common.EkEnumeration.QueryStringActionType.Resolve;
                else
```

```
                    aliasData.QueryStringAction =
Ektron.Cms.Common.EkEnumeration.QueryStringActionType.None;

                aliasData.QueryString = uxQueryString.Text;
                aliasData.IsDefault = uxIsDefault.Checked;
                aliasData.IsEnabled = uxIsEnabled.Checked;

                aliasData = aliasManager.Add(aliasData);
                MessageUtilities.UpdateMessage(uxMessage, "AliasData is added with Id :
" + aliasData.Id.ToString(), Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes an AliasData object from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias ID

Parameters

- `id`. Unique ID of the alias.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasIdLabel" AssociatedControlID="uxAliasId"
```

```
CssClass="span-3 last"
            runat="server" Text="* Alias Id:" />
        <ektronUI:TextField ID="uxAliasId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasManager aliasManager = new AliasManager();
            long aliasId = 0;
            long.TryParse(uxAliasId.Text, out aliasId);
            if (aliasId > 0)
            {
                AliasData aliasData = aliasManager.GetItem(aliasId);
                if (aliasData != null && aliasData.Id > 0)
                {
                    aliasManager.Delete(aliasData.Id);
                    MessageUtilities.UpdateMessage(uxMessage, "AliasData with Id " +
aliasData.Id.ToString() + " has been Deleted.", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
AliasId.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid AliasId.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetAlias

```
GetAlias(System.Int64,System.Int32,Ektron.Cms.Common.EkEnumeration.TargetType)
```

Gets alias by TargetId, LanguageId and TargetType.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Target ID
- * Language ID
- * Target Type

Parameters

- `TargetId`. ID of the target.
- `LanguageId`. ID of Lanaguage.
- `targetType`. Type of target.

Returns

Returns AliasData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxTargetIdLabel" AssociatedControlID="uxTargetId"
CssClass="span-3 last" runat="server" Text="* Target Id:" />
        <ektronUI:TextField  ID="uxTargetId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageListLabel" AssociatedControlID="uxLanguageList"
CssClass="span-3 last" runat="server" Text="* Language Id:" />
        <asp:DropDownList ID="uxLanguageList" runat="server">
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTargetTypeLabel" AssociatedControlID="uxTargetType"
```

```
CssClass="span-3 last" runat="server" Text="* Target Type:" />
        <asp:DropDownList ID="uxTargetType" runat="server">
        <asp:ListItem>Select</asp:ListItem>
        <asp:ListItem>Content</asp:ListItem>
        <asp:ListItem>Group</asp:ListItem>
        <asp:ListItem>User</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Alias"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAliasId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAliasUrl" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTargetQuickLink" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsDefault" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            AliasManager aliasManager = new AliasManager();
            long targetId = 0;
            long.TryParse(uxTargetId.Text, out targetId);
            if (targetId > 0 && uxTargetType.SelectedItem.Text != "Select")
            {
                EkEnumeration.TargetType targetType;
                if(uxTargetType.SelectedItem.Text == "Content")
                    targetType = EkEnumeration.TargetType.Content;
                else if(uxTargetType.SelectedItem.Text == "Group")
                    targetType = EkEnumeration.TargetType.Group;
                else
                    targetType = EkEnumeration.TargetType.User;

                AliasData aliasData = aliasManager.GetAlias(targetId, int.Parse
(uxLanguageList.SelectedItem.Value), targetType);

                if (aliasData != null && aliasData.Id != 0)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Details for Alias with
TargetId " + targetId.ToString() + " are below", Message.DisplayModes.Success);
                    uxAliasId.Text = "Alias Id : " + aliasData.Id.ToString();
                    uxAliasUrl.Text = "Alias Url : " + aliasData.Alias.ToString();
                    uxTargetQuickLink.Text = "Target QuickLink : " +
aliasData.TargetURL.ToString();
                    uxIsEnabled.Text = "IsEnabled : " + aliasData.IsEnabled.ToString();
                    uxIsDefault.Text = "IsDefault : " + aliasData.IsDefault.ToString();
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values
for required fields.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves the properties of a specific alias object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias ID

Parameters

- `Alias id`. Unique ID of the alias.

Returns

AliasData data.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasIdLabel" AssociatedControlID="uxAliasId"
CssClass="span-3 last"
            runat="server" Text="* Alias Id:" />
        <ektronUI:TextField ID="uxAliasId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTargetId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxTargetQuickLink" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLanguageId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsDefault" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasManager aliasManager = new AliasManager();
            long aliasId = 0;
            long.TryParse(uxAliasId.Text, out aliasId);
            if (aliasId != 0)
            {
                AliasData aliasData = aliasManager.GetItem(aliasId);
                if (aliasData != null && aliasData.Id != 0)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Details for Alias with Id
" + aliasData.Id.ToString() + " are below", Message.DisplayModes.Success);
                    uxTargetId.Text = "Target Id : " + aliasData.TargetId.ToString();

                    uxTargetQuickLink.Text = "Target QuickLink : " +
aliasData.TargetURL.ToString();
                    uxLanguageId.Text = "Language Id : " + aliasData.LanguageId.ToString
();
                    uxIsEnabled.Text = "IsEnabled : " + aliasData.IsEnabled.ToString();
                    uxIsDefault.Text = "IsDefault : " + aliasData.IsDefault.ToString();
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
AliasId.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid AliasId.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Alias Property
- * Object Value

Parameters

- `criteria`. AliasCriteria defining which AutoAlias to get.

Returns

List of AliasData data.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasPropertyLabel" AssociatedControlID="uxAliasProperty"
CssClass="span-6 last" runat="server" Text=" Alias Property:" />
        <asp:DropDownList ID="uxAliasProperty" runat="server">
            <asp:ListItem>Alias</asp:ListItem>
            <asp:ListItem>AliasId</asp:ListItem>
            <asp:ListItem>TargetId</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:ListView ID="uxAliasListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
```

```
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        AliasId
                    </th>
                    <th>
                        Alias
                    </th>
                    <th>
                        TargetId
                    </th>
                    <th>
                        TargetURL
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Alias")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TargetId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TargetURL")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasManager aliasManager = new AliasManager();
            string aliasProperty = uxAliasProperty.SelectedItem.Text;
            string objectValue = uxObjectValue.Text;

            AliasCriteria criteria = new AliasCriteria();
            criteria.OrderByField = AliasProperty.Id;
            criteria.OrderByDirection = EkEnumeration.OrderByDirection.Ascending;

            if (aliasProperty == "Alias")
                criteria.AddFilter(AliasProperty.Alias, CriteriaFilterOperator.Contains,
objectValue);
            else if(aliasProperty == "AliasId")
                criteria.AddFilter(AliasProperty.Id, CriteriaFilterOperator.EqualTo,
objectValue);
            else
                criteria.AddFilter(AliasProperty.TargetId,
CriteriaFilterOperator.EqualTo, objectValue);

            List<AliasData> aliasDataList = aliasManager.GetList(criteria);
            uxAliasListView.Visible = true;
            uxAliasListView.DataSource = aliasDataList;
            uxAliasListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTarget

```
GetTarget(System.Uri,System.Int64)
```

Retrieves the alias information for given URL.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Name
- * Extension

- * Language ID

Parameters

- `url`. URL to get alias information.
- `Language ID`. ID of language.

Returns

AliasData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasNameLabel" AssociatedControlID="uxAliasName"
CssClass="span-3 last"
            runat="server" Text="* Alias Name:" />
        <ektronUI:TextField ID="uxAliasName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionLabel" AssociatedControlID="uxExtension"
CssClass="span-3 last" runat="server" Text="* Extension:" />
        <asp:DropDownList ID="uxExtension" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem>.aspx</asp:ListItem>
            <asp:ListItem>/</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageListLabel" AssociatedControlID="uxLanguageList"
CssClass="span-3 last" runat="server" Text="* Language Id:" />
        <asp:DropDownList ID="uxLanguageList" runat="server">
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Target"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentQuickLink" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
```

```
        <asp:Literal ID="uxIsEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsDefault" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasManager aliasManager = new AliasManager();
            if (uxAliasName.Text != "" && uxExtension.SelectedItem.Text != "Select")
            {
                Uri uri = new Uri(Request.Url.Scheme + "://" + Request.Url.Authority +
"/" + uxAliasName.Text + uxExtension.SelectedItem.Text);
                AliasData aliasData = aliasManager.GetTarget(uri, int.Parse
(uxLanguageList.SelectedItem.Value));
                if (aliasData != null && aliasData.Id > 0)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Details for Alias with ID
" + aliasData.Id.ToString() + " are below", Message.DisplayModes.Success);
                    uxContentId.Text = "Content Id : " + aliasData.TargetId.ToString();
                    uxContentQuickLink.Text = "Content QuickLink : " +
aliasData.TargetURL.ToString();
                    uxIsEnabled.Text = "IsEnabled : " + aliasData.IsEnabled.ToString();
                    uxIsDefault.Text = "IsDefault : " + aliasData.IsDefault.ToString();
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values
for required fields.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
```

```
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

IsValidEktronAlias

```
IsValidEktronAlias(System.String)
```

Determines whether an alias name is valid.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Name

Parameters

- `AliasName`. Name of the alias.

Returns

Returns true if valid.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasNameLabel" AssociatedControlID="uxAliasName"
CssClass="span-3 last"
            runat="server" Text="* Alias Name:" />
        <ektronUI:TextField ID="uxAliasName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Target"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasManager aliasManager = new AliasManager();
            string aliasName = uxAliasName.Text;
            if (aliasName != "")
            {
                bool validAlias = aliasManager.IsValidEktronAlias(aliasName);
                if(validAlias)
                    MessageUtilities.UpdateMessage(uxMessage, "AliasName " + aliasName +
" is Valid", Message.DisplayModes.Success);
                else
                    MessageUtilities.UpdateMessage(uxMessage, "AliasName " + aliasName +
" is InValid.", Message.DisplayModes.Error);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid value for
AliasName.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasData)
```

Updates an AliasData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Id
- * Alias Name

- Extension
- Content ID
- Query String
- Is Enabled
- Is Default

Parameters

- `data`. UrlAutoAlias data

Returns

Returns the custom AliasData object updated.

Remarks

Validate the message item with `GetItem()` before you update the properties. Then, call `Update` with your modified AliasData object.

NOTE: You cannot change all properties after the initial Add event, such as the AliasData .Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasIdLabel" AssociatedControlID="uxAliasId"
CssClass="span-3 last" runat="server" Text="* Alias Id:" />
        <ektronUI:TextField  ID="uxAliasId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasNameLabel" AssociatedControlID="uxAliasName"
CssClass="span-3 last" runat="server" Text=" Alias Name:" />
        <ektronUI:TextField  ID="uxAliasName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionLabel" AssociatedControlID="uxExtension"
CssClass="span-3 last" runat="server" Text=" Extension:" />
        <asp:DropDownList ID="uxExtension" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem>.aspx</asp:ListItem>
            <asp:ListItem>/</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text=" Content Id:" />
        <ektronUI:TextField  ID="uxContentId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQueryStringActionLabel"
```

```
AssociatedControlID="uxQueryStringAction" CssClass="span-3 last" runat="server" Text="
QueryString Action:" />
        <asp:DropDownList ID="uxQueryStringAction" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem>None</asp:ListItem>
            <asp:ListItem>Append</asp:ListItem>
            <asp:ListItem>Replace</asp:ListItem>
            <asp:ListItem>Resolve</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQueryStringLabel" AssociatedControlID="uxQueryString"
CssClass="span-3 last" runat="server" Text=" Query String:" />
        <ektronUI:TextField  ID="uxQueryString" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-3 last" runat="server" Text=" Is Enabled:" />
        <asp:checkbox ID="uxIsEnabled" runat="server" Checked="true" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsDefaultLabel" AssociatedControlID="uxIsDefault"
CssClass="span-3 last" runat="server" Text=" Is Default:" />
        <asp:checkbox ID="uxIsDefault" runat="server" Checked="true" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasManager aliasManager = new AliasManager();
            long aliasId = 0;
            long.TryParse(uxAliasId.Text, out aliasId);
```

```
            string queryStringAction = uxQueryStringAction.SelectedItem.Text;
            if (aliasId > 0)
            {
                AliasData aliasData = aliasManager.GetItem(aliasId);
                if (aliasData != null && aliasData.Id > 0)
                {
                    if (uxAliasName.Text != "" || uxExtension.SelectedItem.Text !=
"Select")
                    {
                        string aliasUrl = aliasData.Alias;
                        string extension = "";

                        if (aliasUrl.Contains(".aspx"))
                        {
                            extension = ".aspx";
                            aliasUrl = aliasUrl.Replace(".aspx", "");
                        }
                        else if (aliasUrl.LastIndexOf("/") != -1)
                        {
                            extension = "/";
                            aliasUrl = aliasUrl.Remove(aliasUrl.LastIndexOf("/"), 1);
                        }

                        if (uxAliasName.Text != "")
                            aliasUrl = uxAliasName.Text;

                        if (uxExtension.SelectedItem.Text != "Select")
                            extension = uxExtension.SelectedItem.Text;

                        aliasData.Alias = aliasUrl + extension;
                    }

                    if (uxContentId.Text != "")
                    {
                        if (System.Text.RegularExpressions.Regex.IsMatch
(uxContentId.Text, @"^\d+$"))
                        {
                            Ektron.Cms.Framework.Content.ContentManager contentManager =
new Ektron.Cms.Framework.Content.ContentManager();
                            ContentData contentData = contentManager.GetItem(long.Parse
(uxContentId.Text));
                            if (contentData != null && contentData.Id > 0)
                                aliasData.TargetId = contentData.Id;
                            else
                            {
                                MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid ContentId.", Message.DisplayModes.Error);
                                uxPageMultiView.SetActiveView(uxViewMessage);
                                return;
                            }
                        }
                        else
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid ContentId.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
                            return;
                        }
                    }

                    if (queryStringAction == "Append")
                        aliasData.QueryStringAction =
Ektron.Cms.Common.EkEnumeration.QueryStringActionType.Append;
                    else if (queryStringAction == "Replace")
                        aliasData.QueryStringAction =
Ektron.Cms.Common.EkEnumeration.QueryStringActionType.Replace;
                    else if (queryStringAction == "Resolve")
                        aliasData.QueryStringAction =
Ektron.Cms.Common.EkEnumeration.QueryStringActionType.Resolve;
                    else if (queryStringAction == "None")
                        aliasData.QueryStringAction =
Ektron.Cms.Common.EkEnumeration.QueryStringActionType.None;

                    aliasData.QueryString = uxQueryString.Text != "" ?
uxQueryString.Text : aliasData.QueryString;
                    aliasData.IsDefault = uxIsDefault.Checked;
                    aliasData.IsEnabled = uxIsEnabled.Checked;

                    aliasData = aliasManager.Update(aliasData);
                    MessageUtilities.UpdateMessage(uxMessage, "AliasData is updated with
Id : " + aliasData.Id.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
AliasId.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid AliasId.",
Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

AliasCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Settings.UrlAliasing.DataObjects
```

Constructors

- `AliasCriteria()`

  ```
  public AliasCriteria()
  ```

- `AliasCriteria (Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasProperty, EkEnumeration.OrderByDirection)`

  ```
  public AliasCriteria(Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `AliasProperty` are:

  - Alias
  - ConfigurationId
  - Id
  - IsDefault
  - IsEnabled
  - LanguageId
  - NodeId
  - QueryString
  - QueryStringAction
  - SiteId
  - Target
  - TargetId
  - Type

AliasData

Namespace

```
Ektron.Cms.Settings.UrlAliasing.DataObjects
```

Properties

- `Alias`. The Alias URL, will be used to match incoming requests

  ```
  public string Alias { set; get; }
  ```

- `AliasData()`

  ```
  public AliasData()
  ```

- `ConfigurationId`. Parent Alias configuration that generated this alias.

  ```
  public long ConfigurationId { set; get; }
  ```

- `Id`

  ```
  public long Id { set; get; }
  ```

- `IsDefault`

```
public bool IsDefault { set; get; }
```

- `IsEnabled`

```
public bool IsEnabled { set; get; }
```

- `LanguageId`. The Language Id of the alias.

```
public int LanguageId { set; get; }
```

- `NodeId`

```
public long NodeId { set; get; }
```

- `QueryString`

```
public string QueryString { set; get; }
```

- `QueryStringAction`

```
public EkEnumeration.QueryStringActionType QueryStringAction { set; get; }
```

- `SiteId`. The multisite ID.

```
public long SiteId { set; get; }
```

- `TargetId`

```
public long TargetId { set; get; }
```

- `TargetType`

```
public EkEnumeration.TargetType TargetType { set; get; }
```

- `TargetURL`

```
public string TargetURL { set; get; }
```

- `Type`

```
public EkEnumeration.AliasRuleType Type { set; get; }
```

AliasRuleManager

8.61 and higher

The AliasRuleManager class manages alias rules. See Creating User-Friendly URLs with Aliasing in the Ektron Reference for information about aliases.

Namespace

```
Ektron.Cms.Framework.Settings.UrlAliasing.AliasRuleManager
```

Constructors

- `AliasRuleManager()`
- `AliasRuleManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.

- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Common.UrlAliasAutoData)
```

Adds an alias rule. AliasRuleData.Id is populated with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Rule Type
- * Alias Rule Name
- Language ID
- * Source ID
- Exclude Path ID
- Alias Format
- * Extension

- Query String
- Is Enabled

Parameters

- `data`. AliasRuleData data.

Returns

Returns added AliasRuleData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasRuleTypeLabel" AssociatedControlID="uxAliasRuleType"
CssClass="span-3 last" runat="server" Text="* AliasRule Type:" />
        <asp:DropDownList ID="uxAliasRuleType" runat="server"
OnSelectedIndexChanged="uxAliasRuleType_OnSelectedIndexChanged" AutoPostBack="true">
            <asp:ListItem>Folder</asp:ListItem>
            <asp:ListItem>Group</asp:ListItem>
            <asp:ListItem>Taxonomy</asp:ListItem>
            <asp:ListItem>User</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasRuleNameLabel" AssociatedControlID="uxAliasRuleName"
CssClass="span-3 last" runat="server" Text="* AliasRule Name:" />
        <ektronUI:TextField  ID="uxAliasRuleName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageListLabel" AssociatedControlID="uxLanguageList"
CssClass="span-3 last" runat="server" Text=" Language Id:" />
        <asp:DropDownList ID="uxLanguageList" runat="server" Enabled="false">
            <asp:ListItem>All</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasPrefixLabel" AssociatedControlID="uxAliasPrefix"
CssClass="span-3 last" runat="server" Text=" Alias Prefix:" Visible="false" />
        <ektronUI:TextField  ID="uxAliasPrefix" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  Visible="false"/>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSourceIdLabel" AssociatedControlID="uxSourceId"
CssClass="span-3 last" runat="server" Text="* Source Id:" />
        <ektronUI:TextField  ID="uxSourceId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExcludePathIdLabel" AssociatedControlID="uxExcludePathId"
CssClass="span-3 last" runat="server" Text=" ExcludePath Id:" />
        <ektronUI:TextField  ID="uxExcludePathId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasFormatLabel" AssociatedControlID="uxAliasFormat"
CssClass="span-3 last" runat="server" Text=" Alias Format:" />
        <asp:DropDownList ID="uxAliasFormat" runat="server">
            <asp:ListItem>Content Title</asp:ListItem>
            <asp:ListItem>Content Id</asp:ListItem>
            <asp:ListItem>Content Id and Language</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionLabel" AssociatedControlID="uxExtension"
CssClass="span-3 last" runat="server" Text="* Extension:" />
        <asp:DropDownList ID="uxExtension" runat="server">
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTemplateLabel" AssociatedControlID="uxTemplate"
CssClass="span-3 last" runat="server" Text=" Template:" Visible="false" />
        <asp:DropDownList ID="uxTemplate" runat="server" Visible="false">
            <asp:ListItem>QuickLink</asp:ListItem>
            <asp:ListItem>Taxonomy Template</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQueryStringParamLabel"
AssociatedControlID="uxQueryStringParam" CssClass="span-3 last" runat="server" Text="
Query String Param:" />
        <ektronUI:TextField  ID="uxQueryStringParam" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-3 last" runat="server" Text=" Is Enabled:" />
        <asp:checkbox ID="uxIsEnabled" runat="server" Checked="true" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Organization;
```

```
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasRuleManager aliasRuleManager = new AliasRuleManager();
            string aliasRulename = uxAliasRuleName.Text;
            if (aliasRulename != "")
            {
                AliasRuleData aliasRuleData = new AliasRuleData();
                aliasRuleData.Name = aliasRulename;
                string ruleType = uxAliasRuleType.SelectedItem.Text;
                if (ruleType == "Folder")
                {
                    aliasRuleData.Type = EkEnumeration.AliasRuleType.Folder;
                    FolderManager folderManager = new FolderManager();
                    long folderId = 0;
                    long.TryParse(uxSourceId.Text, out folderId);
                    FolderData folderData = folderManager.GetItem(folderId);
                    if (folderData != null && folderData.Id > 0)
                    {
                        aliasRuleData.SourceId = folderData.Id;
                    }
                    else
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
SourceId.", Message.DisplayModes.Error);
                        uxPageMultiView.SetActiveView(uxViewMessage);
                        return;
                    }

                    List<long> parentFolderIds = new List<long>();
                    long parentId = folderData.Id;
                    folderData = null;
                    long excludePathId = 0;
                    long.TryParse(uxExcludePathId.Text, out excludePathId);
                    if (excludePathId > 0)
                    {
                        folderData = folderManager.GetItem(excludePathId);
                        if (folderData != null && folderData.Id > 0)
                        {
                            while (parentId != 0)
                            {
                                folderData = null;
                                folderData = folderManager.GetItem(parentId);
                                parentFolderIds.Add(folderData.ParentId);
                                parentId = folderData.ParentId;
                            }

                            if (parentFolderIds.Contains(excludePathId))
                                aliasRuleData.ExcludedPathId = excludePathId;
                            else
```

```
                            {
                                MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid ExcludePathId.", Message.DisplayModes.Error);
                                uxPageMultiView.SetActiveView(uxViewMessage);
                                return;
                            }
                        }
                        else
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid ExcludePathId.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
                            return;
                        }
                    }

                    string aliasFormat = uxAliasFormat.SelectedValue;
                    if (aliasFormat == "Content Title")
                        aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentTitle;
                    else if (aliasFormat == "Content Id")
                        aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentId;
                    else
                        aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentIdAndLanguage;

                }
                else if (ruleType == "Group")
                {
                    aliasRuleData.Type = EkEnumeration.AliasRuleType.Group;
                    aliasRuleData.Prefix = uxAliasPrefix.Text;
                    aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentTitle;
                }
                else if (ruleType == "User")
                {
                    aliasRuleData.Type = EkEnumeration.AliasRuleType.User;
                    aliasRuleData.Prefix = uxAliasPrefix.Text;
                    aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentTitle;
                }
                else
                {
                    aliasRuleData.Type = EkEnumeration.AliasRuleType.Taxonomy;
                    aliasRuleData.LanguageId = int.Parse
(uxLanguageList.SelectedItem.Value);

                    TaxonomyManager taxonomyManager = new TaxonomyManager();
                    long taxonomyId = 0;
                    long.TryParse(uxSourceId.Text, out taxonomyId);
                    TaxonomyData taxonomyData = taxonomyManager.GetItem(taxonomyId);
                    if (taxonomyData != null && taxonomyData.Id > 0)
                        aliasRuleData.SourceId = taxonomyData.Id;
                    else
                    {
```

```
                        MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
SourceId.", Message.DisplayModes.Error);
                        uxPageMultiView.SetActiveView(uxViewMessage);
                        return;
                    }

                    List<long> parentTaxonomyIds = new List<long>();
                    long parentId = taxonomyData.Id;
                    taxonomyData = null;
                    long excludePathId = 0;
                    long.TryParse(uxExcludePathId.Text, out excludePathId);
                    if (excludePathId > 0)
                    {
                        taxonomyData = taxonomyManager.GetItem(excludePathId);
                        if (taxonomyData != null && taxonomyData.Id > 0)
                        {
                            while (parentId != 0)
                            {
                                taxonomyData = null;
                                taxonomyData = taxonomyManager.GetItem(parentId);
                                parentTaxonomyIds.Add(taxonomyData.ParentId);
                                parentId = taxonomyData.ParentId;
                            }

                            if (parentTaxonomyIds.Contains(excludePathId))
                                aliasRuleData.ExcludedPathId = excludePathId;
                            else
                            {
                                MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid ExcludePathId.", Message.DisplayModes.Error);
                                uxPageMultiView.SetActiveView(uxViewMessage);
                                return;
                            }
                        }
                        else
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid ExcludePathId.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
                            return;
                        }
                    }

                    string aliasFormat = uxAliasFormat.SelectedValue;
                    if (aliasFormat == "Content Title")
                        aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentTitle;
                    else if (aliasFormat == "Content Id")
                        aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentId;
                    else
                        aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentIdAndLanguage;

                    string template = uxTemplate.SelectedValue;
                    if (template == "QuickLink")
```

```
                        aliasRuleData.TargetTemplateSource =
EkEnumeration.TargetTemplateSource.Quicklink;
                    else
                        aliasRuleData.TargetTemplateSource =
EkEnumeration.TargetTemplateSource.Taxonomy;
                }

                aliasRuleData.FileExtensionId = long.Parse(uxExtension.SelectedValue);
                aliasRuleData.SourceParamName = uxQueryStringParam.Text;
                aliasRuleData.IsEnabled = uxIsEnabled.Checked;

                aliasRuleData = aliasRuleManager.Add(aliasRuleData);
                MessageUtilities.UpdateMessage(uxMessage, "AliasRuleData is added with
Id " + aliasRuleData.Id.ToString(), Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid AliasRule
Name.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes alias rule and aliases by configuration ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Rule ID

Parameters

- `ConfigId`. ID of AliasRuleData configuration.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasRuleIdLabel" AssociatedControlID="uxAliasRuleId"
CssClass="span-3 last" runat="server" Text="* AliasRule Id:" />
        <ektronUI:TextField  ID="uxAliasRuleId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasRuleManager aliasRuleManager = new AliasRuleManager();
            long aliasRuleId = 0;
            long.TryParse(uxAliasRuleId.Text, out aliasRuleId);
            AliasRuleData aliasRuleData = aliasRuleManager.GetItem(aliasRuleId);
            if (aliasRuleData != null && aliasRuleData.Id != 0)
            {
                aliasRuleManager.Delete(aliasRuleData.Id);
                MessageUtilities.UpdateMessage(uxMessage, "AliasRuleData with Id " +
aliasRuleData.Id.ToString() + " has been Deleted.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid AliasRule
Id.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeleteAliases

```
DeleteAliases(System.Int64)
```

Delete aliases by configuration ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Rule ID

Parameters

- `ConfigId`. ID of AliasRuleData configuration.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasRuleIdLabel" AssociatedControlID="uxAliasRuleId"
CssClass="span-3 last" runat="server" Text="* AliasRule Id:" />
        <ektronUI:TextField  ID="uxAliasRuleId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete Aliases"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasRuleManager aliasRuleManager = new AliasRuleManager();
            long aliasRuleId = 0;
            long.TryParse(uxAliasRuleId.Text, out aliasRuleId);
            AliasRuleData aliasRuleData = aliasRuleManager.GetItem(aliasRuleId);
            if (aliasRuleData != null && aliasRuleData.Id != 0)
            {
                aliasRuleManager.DeleteAliases(aliasRuleData.Id);
                MessageUtilities.UpdateMessage(uxMessage, "Aliases with AliasRule Id " +
aliasRuleData.Id.ToString() + " has been Deleted.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid AliasRule
Id.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DisableAliases

```
DisableAliases(System.Int64)
```

Disables aliases by ConfigId.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Rule ID

Parameters

- `ConfigId`. ID of AliasRuleData configuration.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxAliasRuleIdLabel" AssociatedControlID="uxAliasRuleId"
CssClass="span-3 last" runat="server" Text="* AliasRule Id:" />
        <ektronUI:TextField  ID="uxAliasRuleId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Disable Aliases"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasRuleManager aliasRuleManager = new AliasRuleManager();
            long aliasRuleId = 0;
            long.TryParse(uxAliasRuleId.Text, out aliasRuleId);
            AliasRuleData aliasRuleData = aliasRuleManager.GetItem(aliasRuleId);
            if (aliasRuleData != null && aliasRuleData.Id != 0)
            {
                aliasRuleManager.DisableAliases(aliasRuleData.Id);
                MessageUtilities.UpdateMessage(uxMessage, "Aliases with AliasRule Id " +
aliasRuleData.Id.ToString() + " has been Disabled.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid AliasRule
Id.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DisableAll

```
DisableAll()
```

Disables all aliases.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Disable All"></ektronUI:Button>
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasRuleManager aliasRuleManager = new AliasRuleManager();
            aliasRuleManager.DisableAll();
            MessageUtilities.UpdateMessage(uxMessage, "All AliasRules has been
Disabled.", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

EnableAliases

```
EnableAliases(System.Int64)
```

Enables aliases by ConfigId.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Rule ID

Parameters

- `ConfigId`. ID of AliasRuleData configuration.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasRuleIdLabel" AssociatedControlID="uxAliasRuleId"
CssClass="span-3 last" runat="server" Text="* AliasRule Id:" />
        <ektronUI:TextField  ID="uxAliasRuleId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Enable Aliases"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasRuleManager aliasRuleManager = new AliasRuleManager();
            long aliasRuleId = 0;
            long.TryParse(uxAliasRuleId.Text, out aliasRuleId);
            AliasRuleData aliasRuleData = aliasRuleManager.GetItem(aliasRuleId);
            if (aliasRuleData != null && aliasRuleData.Id != 0)
```

```
            {
                aliasRuleManager.EnableAliases(aliasRuleData.Id);
                MessageUtilities.UpdateMessage(uxMessage, "Aliases with AliasRule Id " +
aliasRuleData.Id.ToString() + " has been Enabled.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid AliasRule
Id.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves AliasRuleData object by ConfigId.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Rule ID

Parameters

- `ConfigId`. ID of AliasRuleData configuration.

Returns

AliasRuleData data object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasRuleIdLabel" AssociatedControlID="uxAliasRuleId"
CssClass="span-3 last" runat="server" Text="* AliasRule Id:" />
        <ektronUI:TextField  ID="uxAliasRuleId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
```

```
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAliaRuleName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAliasRuleType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSourceId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxExcludePathId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAliasPrefix" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasRuleManager aliasRuleManager = new AliasRuleManager();
            long aliasRuleId = 0;
            long.TryParse(uxAliasRuleId.Text, out aliasRuleId);
            AliasRuleData aliasRuleData = aliasRuleManager.GetItem(aliasRuleId);
            if (aliasRuleData != null && aliasRuleData.Id != 0)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for AliasRule with Id
" + aliasRuleData.Id.ToString() + " are below", Message.DisplayModes.Success);
                uxAliaRuleName.Text = "AliasRule Name : " + aliasRuleData.Name.ToString
();
                uxAliasRuleType.Text = "AliasRule Type : " + aliasRuleData.Type.ToString
();
                uxSourceId.Text = "Source Id : " + aliasRuleData.SourceId.ToString();
```

```
                uxExcludePathId.Text = "ExcludedPath Id : " +
aliasRuleData.ExcludedPathId.ToString();
                uxAliasPrefix.Text = "Alias Prefix : " + aliasRuleData.Prefix.ToString
();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid AliasRule
Id.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (AliasRule)

```
GetList(Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasRuleCriteria)
```

Retrieves list of AliasRuleData objects based on supplied criteria.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Rule Property
- * Object Value

Parameters

- `criteria`. AliasRuleCriteria used to retrieve the AliasRule.

Returns

Returns list of AliasRuleData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasRulePropertyLabel"
AssociatedControlID="uxAliasRuleProperty" CssClass="span-6 last" runat="server" Text="*
AliasRule Property:" />
        <asp:DropDownList ID="uxAliasRuleProperty" runat="server">
```

```
            <asp:ListItem>AliasRuleId</asp:ListItem>
            <asp:ListItem>AliasRuleName</asp:ListItem>
            <asp:ListItem>SourceId</asp:ListItem>
            <asp:ListItem>AliasPrefix</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:ListView ID="uxAliasRuleListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        AliasRuleId
                    </th>
                    <th>
                        Name
                    </th>
                    <th>
                        SourceId
                    </th>
                    <th>
                        ExcludedPathId
                    </th>
                    <th>
                        Prefix
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
```

```
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("SourceId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ExcludedPathId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Prefix")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasRuleManager aliasRuleManager = new AliasRuleManager();
            string aliasRuleProperty = uxAliasRuleProperty.SelectedItem.Text;
            string objectValue = uxObjectValue.Text;

            AliasRuleCriteria criteria = new AliasRuleCriteria();
            criteria.OrderByField = AliasRuleProperty.Id;
            criteria.OrderByDirection = EkEnumeration.OrderByDirection.Ascending;

            if (aliasRuleProperty == "AliasRuleId")
                criteria.AddFilter(AliasRuleProperty.Id,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, objectValue);
            else if (aliasRuleProperty == "AliasRuleName")
                criteria.AddFilter(AliasRuleProperty.Name,
Ektron.Cms.Common.CriteriaFilterOperator.Contains, objectValue);
            else if (aliasRuleProperty == "SourceId")
                criteria.AddFilter(AliasRuleProperty.SourceId,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, objectValue);
            else
                criteria.AddFilter(AliasRuleProperty.Prefix,
Ektron.Cms.Common.CriteriaFilterOperator.Contains, objectValue);

            List<AliasRuleData> aliasRuleDataList = aliasRuleManager.GetList(criteria);
            uxAliasRuleListView.Visible = true;
```

```
            uxAliasRuleListView.DataSource = aliasRuleDataList;
            uxAliasRuleListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList (objects)

```
GetList
(System.Int64,Ektron.Cms.Common.EkEnumeration.AliasRuleType,System.Int64,System.Int32)
```

Retrieves list of AliasRuleData objects by SourceId, AliasRuleType, SiteId and LanguageId.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Source ID
- * Alias Rule Type
- * Site ID
- * Language ID

Parameters

- `SourceId`. ID of the source, for that AliasRule is created.
- `type`. Type of AliasRule.
- `siteId`. ID of the site.
- `languageId`. ID of the language.

Returns

Returns list of AliasRuleData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxSourceIdLabel" AssociatedControlID="uxSourceId"
CssClass="span-3 last" runat="server" Text="* Source Id:" />
```

```
        <ektronUI:TextField  ID="uxSourceId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasRuleTypeLabel" AssociatedControlID="uxAliasRuleType"
CssClass="span-3 last" runat="server" Text="* AliasRule Type:" />
        <asp:DropDownList ID="uxAliasRuleType" runat="server">
            <asp:ListItem>Folder</asp:ListItem>
            <asp:ListItem>Taxonomy</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSiteIdLabel" AssociatedControlID="uxSiteId"
CssClass="span-3 last" runat="server" Text="* Site Id:" />
        <ektronUI:TextField  ID="uxSiteId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLanguageListLabel" AssociatedControlID="uxLanguageList"
CssClass="span-3 last" runat="server" Text="* Language Id:" />
        <asp:DropDownList ID="uxLanguageList" runat="server">
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxAliasRuleListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >No records found for given inputs.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        AliasRuleId
                    </th>
                    <th>
                        Name
                    </th>
                    <th>
                        SourceId
                    </th>
                    <th>
                        ExcludedPathId
                    </th>
                    <th>
                        Prefix
                    </th>
                </tr>
            </thead>
```

```
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Name")%>
            </td>
            <td class="devsite-method">
                <%# Eval("SourceId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ExcludedPathId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Prefix")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasRuleManager aliasRuleManager = new AliasRuleManager();
            long sourceId = 0;
            long.TryParse(uxSourceId.Text, out sourceId);
            if (sourceId > 0)
            {
                string aliasRuleType = uxAliasRuleType.SelectedItem.Text;
                if (aliasRuleType == "Folder")
                {
                    FolderManager folderManager = new FolderManager();
```

```
                    FolderData folderData = folderManager.GetItem(sourceId);
                    if (!(folderData != null && folderData.Id > 0))
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
SourceId.", Message.DisplayModes.Error);
                        uxPageMultiView.SetActiveView(uxViewMessage);
                        return;
                    }
                }
                else
                {
                    TaxonomyManager taxonomyManager = new TaxonomyManager();
                    TaxonomyData taxonomyData = taxonomyManager.GetItem(sourceId);
                    if (!(taxonomyData != null && taxonomyData.Id > 0))
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
SourceId.", Message.DisplayModes.Error);
                        uxPageMultiView.SetActiveView(uxViewMessage);
                        return;
                    }
                }

                if (!System.Text.RegularExpressions.Regex.IsMatch(uxSiteId.Text,
@"^\d+$"))
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
SiteId.", Message.DisplayModes.Error);
                    uxPageMultiView.SetActiveView(uxViewMessage);
                    return;
                }

                List<AliasRuleData> aliasRuleDataList = aliasRuleManager.GetList
(sourceId, aliasRuleType == "Folder" ? EkEnumeration.AliasRuleType.Folder :
EkEnumeration.AliasRuleType.Taxonomy, long.Parse(uxSiteId.Text), int.Parse
(uxLanguageList.SelectedValue));
                uxAliasRuleListView.Visible = true;
                uxAliasRuleListView.DataSource = aliasRuleDataList;
                uxAliasRuleListView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
SourceId.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasRuleData)
```

Updates the supplied AliasRuleData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Rule ID
- Alias Rule Name
- Source ID
- Exclude Path ID
- Alias Format
- Extension
- Query String
- Is Enabled

Parameters

- `data`. AliasRuleData object.

Returns

Returns updated AliasRuleData object.

Remarks

Validate the message item with `GetItem()` before you update the properties. Then, call `Update` with your modified AliasRuleData object.

NOTE: You cannot change all properties after the initial Add event, such as the UrlAliasAutoData .Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasRuleTypeLabel" AssociatedControlID="uxAliasRuleType"
CssClass="span-3 last" runat="server" Text="AliasRule Type:" />
        <asp:DropDownList ID="uxAliasRuleType" runat="server"
OnSelectedIndexChanged="uxAliasRuleType_OnSelectedIndexChanged" AutoPostBack="true">
            <asp:ListItem Value="2">Folder</asp:ListItem>
            <asp:ListItem Value="5">Group</asp:ListItem>
            <asp:ListItem Value="1">Taxonomy</asp:ListItem>
            <asp:ListItem Value="4">User</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasRuleIdLabel" AssociatedControlID="uxAliasRuleId"
```

```
CssClass="span-3 last" runat="server" Text="* AliasRule Id:" />
        <ektronUI:TextField  ID="uxAliasRuleId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasRuleNameLabel" AssociatedControlID="uxAliasRuleName"
CssClass="span-3 last" runat="server" Text=" AliasRule Name:" />
        <ektronUI:TextField  ID="uxAliasRuleName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasPrefixLabel" AssociatedControlID="uxAliasPrefix"
CssClass="span-3 last" runat="server" Text=" Alias Prefix:" Visible="false" />
        <ektronUI:TextField  ID="uxAliasPrefix" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  Visible="false"/>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSourceIdLabel" AssociatedControlID="uxSourceId"
CssClass="span-3 last" runat="server" Text=" Source Id:" />
        <ektronUI:TextField  ID="uxSourceId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExcludePathIdLabel" AssociatedControlID="uxExcludePathId"
CssClass="span-3 last" runat="server" Text=" ExcludePath Id:" />
        <ektronUI:TextField  ID="uxExcludePathId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasFormatLabel" AssociatedControlID="uxAliasFormat"
CssClass="span-3 last" runat="server" Text=" Alias Format:" />
        <asp:DropDownList ID="uxAliasFormat" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem>Content Title</asp:ListItem>
            <asp:ListItem>Content Id</asp:ListItem>
            <asp:ListItem>Content Id and Language</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionLabel" AssociatedControlID="uxExtension"
CssClass="span-3 last" runat="server" Text=" Extension:" />
        <asp:DropDownList ID="uxExtension" runat="server">
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxTemplateLabel" AssociatedControlID="uxTemplate"
CssClass="span-3 last" runat="server" Text=" Template:" Visible="false" />
        <asp:DropDownList ID="uxTemplate" runat="server" Visible="false">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem>QuickLink</asp:ListItem>
            <asp:ListItem>Taxonomy Template</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQueryStringParamLabel"
AssociatedControlID="uxQueryStringParam" CssClass="span-3 last" runat="server" Text="
```

```
Query String Param:" />
        <ektronUI:TextField  ID="uxQueryStringParam" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-3 last" runat="server" Text=" Is Enabled:" />
        <asp:checkbox ID="uxIsEnabled" runat="server" Checked="true" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Organization;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasRuleManager aliasRuleManager = new AliasRuleManager();
            long aliasRuleId = 0;
            long.TryParse(uxAliasRuleId.Text, out aliasRuleId);

            if (aliasRuleId != 0)
            {
                string ruleType = uxAliasRuleType.SelectedItem.Text;
                AliasRuleData aliasRuleData = aliasRuleManager.GetItem(aliasRuleId);
                if (aliasRuleData != null && aliasRuleData.Id != 0)
                {
                    if (aliasRuleData.Type != (EkEnumeration.AliasRuleType)int.Parse
(uxAliasRuleType.SelectedValue))
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
AliasRule Id.", Message.DisplayModes.Error);
                        uxPageMultiView.SetActiveView(uxViewMessage);
                        return;
                    }
```

```
                    aliasRuleData.Name = uxAliasRuleName.Text != "" ?
uxAliasRuleName.Text : aliasRuleData.Name;
                    if (ruleType == "Folder")
                    {
                        FolderManager folderManager = new FolderManager();

                        FolderData folderData = new FolderData();
                        if (uxSourceId.Text != "")
                        {
                            long folderId = 0;
                            long.TryParse(uxSourceId.Text, out folderId);
                            folderData = folderManager.GetItem(folderId);
                            if (folderData != null && folderData.Id > 0)
                            {
                                aliasRuleData.SourceId = folderData.Id;
                            }
                            else
                            {
                                MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid SourceId.", Message.DisplayModes.Error);
                                uxPageMultiView.SetActiveView(uxViewMessage);
                                return;
                            }
                        }

                        List<long> parentFolderIds = new List<long>();
                        long parentId = folderData.Id;
                        folderData = null;
                        if (uxExcludePathId.Text != "")
                        {
                            long excludePathId = 0;
                            long.TryParse(uxExcludePathId.Text, out excludePathId);
                            folderData = folderManager.GetItem(excludePathId);
                            if (folderData != null && folderData.Id > 0)
                            {
                                while (parentId != 0)
                                {
                                    folderData = null;
                                    folderData = folderManager.GetItem(parentId);
                                    parentFolderIds.Add(folderData.ParentId);
                                    parentId = folderData.ParentId;
                                }

                                if (parentFolderIds.Contains(excludePathId))
                                    aliasRuleData.ExcludedPathId = excludePathId;
                                else
                                {
                                    MessageUtilities.UpdateMessage(uxMessage, "Please
enter valid ExcludePathId.", Message.DisplayModes.Error);
                                    uxPageMultiView.SetActiveView(uxViewMessage);
                                    return;
                                }
                            }
                            else
                            {
```

```
                                MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid ExcludePathId.", Message.DisplayModes.Error);
                                uxPageMultiView.SetActiveView(uxViewMessage);
                                return;
                            }
                        }

                        string aliasFormat = uxAliasFormat.SelectedValue;
                        if (aliasFormat != "Select")
                        {
                            if (aliasFormat == "Content Title")
                                aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentTitle;
                            else if (aliasFormat == "Content Id")
                                aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentId;
                            else
                                aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentIdAndLanguage;
                        }

                    }
                    else if (ruleType == "Group" || ruleType == "User")
                    {
                        aliasRuleData.Prefix = uxAliasPrefix.Text != "" ?
uxAliasPrefix.Text : aliasRuleData.Prefix;
                    }
                    else
                    {
                        TaxonomyManager taxonomyManager = new TaxonomyManager();
                        TaxonomyData taxonomyData = new TaxonomyData();
                        if (uxSourceId.Text != "")
                        {
                            long taxonomyId = 0;
                            long.TryParse(uxSourceId.Text, out taxonomyId);
                            taxonomyData = taxonomyManager.GetItem(taxonomyId);
                            if (taxonomyData != null && taxonomyData.Id > 0)
                                aliasRuleData.SourceId = taxonomyData.Id;
                            else
                            {
                                MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid SourceId.", Message.DisplayModes.Error);
                                uxPageMultiView.SetActiveView(uxViewMessage);
                                return;
                            }
                        }

                        List<long> parentTaxonomyIds = new List<long>();
                        long parentId = taxonomyData.Id;
                        taxonomyData = null;

                        if (uxExcludePathId.Text != "")
                        {
                            long excludePathId = 0;
                            long.TryParse(uxExcludePathId.Text, out excludePathId);
                            taxonomyData = taxonomyManager.GetItem(excludePathId);
```

```
                            if (taxonomyData != null && taxonomyData.Id > 0)
                            {
                                while (parentId != 0)
                                {
                                    taxonomyData = null;
                                    taxonomyData = taxonomyManager.GetItem(parentId);
                                    parentTaxonomyIds.Add(taxonomyData.ParentId);
                                    parentId = taxonomyData.ParentId;
                                }

                                if (parentTaxonomyIds.Contains(excludePathId))
                                    aliasRuleData.ExcludedPathId = excludePathId;
                                else
                                {
                                    MessageUtilities.UpdateMessage(uxMessage, "Please
enter valid ExcludePathId.", Message.DisplayModes.Error);
                                    uxPageMultiView.SetActiveView(uxViewMessage);
                                    return;
                                }
                            }
                            else
                            {
                                MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid ExcludePathId.", Message.DisplayModes.Error);
                                uxPageMultiView.SetActiveView(uxViewMessage);
                                return;
                            }
                        }

                        string aliasFormat = uxAliasFormat.SelectedValue;
                        if (aliasFormat != "Select")
                        {
                            if (aliasFormat == "Content Title")
                                aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentTitle;
                            else if (aliasFormat == "Content Id")
                                aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentId;
                            else
                                aliasRuleData.PageTypeId =
EkEnumeration.AutoAliasNameType.ContentIdAndLanguage;
                        }

                        string template = uxTemplate.SelectedValue;
                        if (template != "Select")
                        {
                            if (template == "QuickLink")
                                aliasRuleData.TargetTemplateSource =
EkEnumeration.TargetTemplateSource.Quicklink;
                            else
                                aliasRuleData.TargetTemplateSource =
EkEnumeration.TargetTemplateSource.Taxonomy;
                        }
                    }

                    aliasRuleData.FileExtensionId = uxExtension.SelectedValue != "0" ?
```

```
long.Parse(uxExtension.SelectedValue) : aliasRuleData.FileExtensionId;
                    aliasRuleData.SourceParamName = uxQueryStringParam.Text != "" ?
uxQueryStringParam.Text : aliasRuleData.SourceParamName;
                    aliasRuleData.IsEnabled = uxIsEnabled.Checked;

                    aliasRuleData = aliasRuleManager.Update(aliasRuleData);
                    MessageUtilities.UpdateMessage(uxMessage, "AliasRuleData is updated
with Id " + aliasRuleData.Id.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
AliasRule Id.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid AliasRule
Id.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

AliasRuleCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Settings.UrlAliasing.DataObjects
```

Constructors

- `AliasRuleCriteria()`

  ```
  public AliasRuleCriteria()
  ```

- `AliasRuleCriteria (Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasRuleProperty, EkEnumeration.OrderByDirection)`

  ```
  public AliasRuleCriteria
  (Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasRuleProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `AliasRuleProperty` are:

- AliasPriority
- ExcludePathId
- FileExtensionId
- Id
- IsDeleted
- IsEnabled
- LanguageId
- Name
- PageTypeId
- Prefix
- SiteId
- SourceId
- SourceParamName
- TargetTemplateSource
- Type

AliasRuleData

Namespace

```
Ektron.Cms.Settings.UrlAliasing.DataObjects
```

Properties

- `AliasPriority`. The priority order.

  ```
  public int AliasPriority { set; get; }
  ```

- `AliasRuleData()`

  ```
  public AliasRuleData()
  ```

- `ExcludePathId`

  ```
  public long ExcludedPathId { set; get; }
  ```

- `FileExtensionId`. The file extension to use in alias generation.

  ```
  public long FileExtensionId { set; get; }
  ```

- `Id`. Unique Identifier for this record.

  ```
  public long Id { set; get; }
  ```

- `IsDeleted`. Indicates deleted status. Rules are soft deleted.

  ```
  public bool IsDeleted { set; get; }
  ```

- `IsEnabled`. Indicates if this configuration is enabled.

  ```
  public bool IsEnabled { set; get; }
  ```

- `LanguageId`. The language ID.

```
public int LanguageId { set; get; }
```

- `Name`. Friendly name of this configuration.

```
public string Name { set; get; }
```

- `PageTypeId`

```
public EkEnumeration.AutoAliasNameType PageTypeId { set; get; }
```

- `Prefix`

```
public string Prefix { set; get; }
```

- `ReplacementCharacterMap`

```
public System.Collections.Generic.List<ReplacementCharacter>
  ReplacementCharacterMap { set; get; }
```

- `SiteId`. The multisite ID.

```
public long SiteId { set; get; }
```

- `SourceId`. Optional. Source entity in the CMS. This changes meaning depending on the ConfigurationType. If its Folder Alias this is a folder Id, if taxonomy alias, its a Taxonomy ID.

```
public long SourceId { set; get; }
```

- `SourceParamName`

```
public string SourceParamName { set; get; }
```

- `TargetTemplateSource`

```
public EkEnumeration.TargetTemplateSource TargetTemplateSource { set; get;
}
```

- `Type`. The type of alias.

```
public EkEnumeration.AliasRuleType Type { set; get; }
```

AliasSettingManager

8.61 and higher

The AliasSettingManager class manages alias settings. See Creating User-Friendly URLs with Aliasing in the Ektron Reference for information about aliases.

Namespace

```
Ektron.Cms.Framework.Settings.UrlAliasing.AliasSettingManager
```

Constructors

- `AliasSettingManager()`
- `AliasSettingManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

AddFileExtension

```
AddFileExtension(Ektron.Cms.Settings.UrlAliasing.DataObjects.FileExtension)
```

Adds a file extension. FileExtension.Id is populated with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * File Extension

Parameters

- `extension`. File extension object.

Returns

Returns FileExtension object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFileExtensionLabel" AssociatedControlID="uxFileExtension"
CssClass="span-3 last" runat="server" Text="* File Extension:" />
        <ektronUI:TextField  ID="uxFileExtension" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Add
FileExtension"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxFileExtensionLabel" AssociatedControlID="uxFileExtension"
CssClass="span-3 last" runat="server" Text="* File Extension:" />
        <ektronUI:TextField  ID="uxFileExtension" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Add
FileExtension"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();

            string fileExtension = uxFileExtension.Text;
            if (fileExtension != "")
            {
                if (Regex.IsMatch(fileExtension, @"[~@#$^*()_+=[\]{}|\\,?: -]"))
```

```
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
FileExtension.", Message.DisplayModes.Error);
                    uxPageMultiView.SetActiveView(uxViewMessage);
                    return;
                }

                List<string> existingFileExtensions =
aliasSettingsManager.GetAllExtensionsOnly();
                if (!existingFileExtensions.Contains(fileExtension))
                {
                    if (!Regex.IsMatch(fileExtension, @"^\."))
                        fileExtension = "." + fileExtension;

                    FileExtension objFileExtension = new FileExtension();
                    objFileExtension.Extension = fileExtension;
                    objFileExtension = aliasSettingsManager.AddFileExtension
(objFileExtension);
                    MessageUtilities.UpdateMessage(uxMessage, "FileExtension is added
with Id : " + objFileExtension.Id.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "FileExtension " +
fileExtension + " already exists.", Message.DisplayModes.Error);
                    uxPageMultiView.SetActiveView(uxViewMessage);
                    return;
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
FileExtension.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

ClearCache

```
ClearCache(System.String)
```

Clears the cache by segment key.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Segment Key

Parameters

- `segmentkey`. Key of segment.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxSegmentKeyLabel" AssociatedControlID="uxSegmentKey"
CssClass="span-4 last" runat="server" Text="* Segment Key:" />
        <ektronUI:TextField  ID="uxSegmentKey" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Clear"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.BusinessObjects.Caching;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();
            string segmentKey = uxSegmentKey.Text;
            if (segmentKey != string.Empty)
            {
                CacheManager cacheManager = new CacheManager();
                object objectCache = cacheManager.Get(segmentKey);
                if (objectCache != null)
                {
                    aliasSettingsManager.ClearCache(segmentKey);
                    MessageUtilities.UpdateMessage(uxMessage, "Cache with SegmentKey " +
segmentKey + " has been Cleared", Message.DisplayModes.Success);
```

```
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
SegmentKey.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
SegmentKey.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeleteExtension

```
DeleteExtension(System.Int64)
```

Deletes an extension by extension ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Extension ID

Parameters

- `extensionid`. ID of the extension.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionIdLabel" AssociatedControlID="uxExtensionId"
CssClass="span-4 last" runat="server" Text="* Extension Id:" />
        <ektronUI:TextField  ID="uxExtensionId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();
            long extensionId = 0;
            long.TryParse(uxExtensionId.Text, out extensionId);
            if (extensionId != 0)
            {
                FileExtension fileExtension = aliasSettingsManager.GetExtension
(extensionId);
                if (fileExtension != null && fileExtension.Id != 0)
                {
                    aliasSettingsManager.DeleteExtension(fileExtension.Id);
                    MessageUtilities.UpdateMessage(uxMessage, "FileExtension with Id " +
fileExtension.Id.ToString() + " has been Deleted.", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
ExtensionId.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
ExtensionId.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Get ()

```
Get()
```

Retrieves AliasSettings object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Get"></ektronUI:Button>
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsAliasingEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsURLRedirectEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxOverrideTemplate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDefaultReplacementCharacter" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();
            AliasSettings aliasSettings = aliasSettingsManager.Get();
            MessageUtilities.UpdateMessage(uxMessage, "Details for AliasSettings with
SiteId " + aliasSettings.SiteID.ToString() + " are below",
Message.DisplayModes.Success);
            uxIsAliasingEnabled.Text = "IsAliasingEnabled : " +
aliasSettings.IsAliasingEnabled.ToString();
            uxIsURLRedirectEnabled.Text = "IsURLRedirectEnabled : " +
aliasSettings.IsURLRedirectEnabled.ToString();
            uxOverrideTemplate.Text = "OverrideTemplate : " +
aliasSettings.OverrideTemplate.ToString();
            uxDefaultReplacementCharacter.Text = "DefaultReplacementCharacter : " +
aliasSettings.DefaultReplacementCharacter.ToString();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Get (name and site)

```
Get(Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasSettingName,System.Int64)
```

Retrieves the AliasSettings value by setting name and site ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Setting Name
- * Site ID

Parameters

- `settingName`. Name of setting.
- `siteID`. ID of site.

Returns

Returns AliasSettings value.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasSettingNameLabel"
AssociatedControlID="uxAliasSettingName" CssClass="span-4 last" runat="server" Text="*
Alias Setting Name:" />
        <asp:DropDownList ID="uxAliasSettingName" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem>DefaultReplaceCharString</asp:ListItem>
            <asp:ListItem>DefaultReplacementCharacter</asp:ListItem>
            <asp:ListItem>IsAliasFallbackEnabled</asp:ListItem>
            <asp:ListItem>IsAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsFolderAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsGroupAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsManualAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsRegExAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsTaxonomyAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsURLRedirectEnabled</asp:ListItem>
            <asp:ListItem>IsUserAliasingEnabled</asp:ListItem>
            <asp:ListItem>OverrideTemplate</asp:ListItem>
            <asp:ListItem>QueryStringAction</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSiteIdLabel" AssociatedControlID="uxSiteId"
CssClass="span-4 last" runat="server" Text="* Site Id:" />
        <ektronUI:TextField  ID="uxSiteId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Get"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Text.RegularExpressions;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
```

```
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();
            string aliasSettingName = uxAliasSettingName.SelectedItem.Text;
            if (aliasSettingName != "Select" && Regex.IsMatch(uxSiteId.Text, @"^\d+$"))
            {
                AliasSettingName settingName;
                if (aliasSettingName == "DefaultReplaceCharString")
                    settingName = AliasSettingName.DefaultReplaceCharString;
                else if (aliasSettingName == "DefaultReplacementCharacter")
                    settingName = AliasSettingName.DefaultReplacementCharacter;
                else if (aliasSettingName == "IsAliasFallbackEnabled")
                    settingName = AliasSettingName.IsAliasFallbackEnabled;
                else if (aliasSettingName == "IsAliasingEnabled")
                    settingName = AliasSettingName.IsAliasingEnabled;
                else if (aliasSettingName == "IsFolderAliasingEnabled")
                    settingName = AliasSettingName.IsFolderAliasingEnabled;
                else if (aliasSettingName == "IsGroupAliasingEnabled")
                    settingName = AliasSettingName.IsGroupAliasingEnabled;
                else if (aliasSettingName == "IsManualAliasingEnabled")
                    settingName = AliasSettingName.IsManualAliasingEnabled;
                else if (aliasSettingName == "IsRegExAliasingEnabled")
                    settingName = AliasSettingName.IsRegExAliasingEnabled;
                else if (aliasSettingName == "IsTaxonomyAliasingEnabled")
                    settingName = AliasSettingName.IsTaxonomyAliasingEnabled;
                else if (aliasSettingName == "IsURLRedirectEnabled")
                    settingName = AliasSettingName.IsURLRedirectEnabled;
                else if (aliasSettingName == "IsUserAliasingEnabled")
                    settingName = AliasSettingName.IsUserAliasingEnabled;
                else if (aliasSettingName == "OverrideTemplate")
                    settingName = AliasSettingName.OverrideTemplate;
                else
                    settingName = AliasSettingName.QueryStringAction;

                string settingValue = aliasSettingsManager.Get(settingName, long.Parse
(uxSiteId.Text));

                if(settingValue != "")
                    MessageUtilities.UpdateMessage(uxMessage, "SettingValue for
SettingName " + aliasSettingName + " and SiteId " + uxSiteId.Text + " is " +
settingValue, Message.DisplayModes.Success);
                else
                    MessageUtilities.UpdateMessage(uxMessage, "SettingValue for
SettingName " + aliasSettingName + " and SiteId " + uxSiteId.Text + " is Not Found.",
Message.DisplayModes.Error);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Get (site)

```
Get(System.Int64)
```

Retrieves an AliasSettings object by site ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Site ID

Parameters

- `siteID`. ID of site.

Returns

Returns AliasSettings object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxSiteIdLabel" AssociatedControlID="uxSiteId"
CssClass="span-4 last" runat="server" Text="* Site Id:" />
        <ektronUI:TextField  ID="uxSiteId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Get"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsAliasingEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsURLRedirectEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
```

```
        <asp:Literal ID="uxOverrideTemplate" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDefaultReplacementCharacter" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Text.RegularExpressions;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();
            if (Regex.IsMatch(uxSiteId.Text, @"^\d+$"))
            {
                AliasSettings aliasSettings = aliasSettingsManager.Get(long.Parse
(uxSiteId.Text));
                if (aliasSettings != null)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Details for AliasSettings
with SiteId " + aliasSettings.SiteID.ToString() + " are below",
Message.DisplayModes.Success);
                    uxIsAliasingEnabled.Text = "IsAliasingEnabled : " +
aliasSettings.IsAliasingEnabled.ToString();
                    uxIsURLRedirectEnabled.Text = "IsURLRedirectEnabled : " +
aliasSettings.IsURLRedirectEnabled.ToString();
                    uxOverrideTemplate.Text = "OverrideTemplate : " +
aliasSettings.OverrideTemplate.ToString();
                    uxDefaultReplacementCharacter.Text = "DefaultReplacementCharacter :
" + aliasSettings.DefaultReplacementCharacter.ToString();
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
SiteId.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid SiteId.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetAllExtensions()

GetAllExtensions()

Retrieves list of FileExtension objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Returns

Returns a list of FileExtension objects.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Get"></ektronUI:Button>
    </li>
</ol>

<asp:ListView ID="uxExtensionListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >No records found for given input.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        FileExtension Id
                    </th>
                    <th>
                        FileExtension
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
```

```
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Extension")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();

            List<FileExtension> fileExtensionList =
aliasSettingsManager.GetAllExtensions();
            uxExtensionListView.Visible = true;
            uxExtensionListView.DataSource = fileExtensionList;
            uxExtensionListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetAllExtensions (ID)

```
GetAllExtensions(System.Int64)
```

Retrieves FileExtension objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Site ID

Parameters

- `siteId`. ID of site.

Returns

Returns FileExtension objects.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionIdLabel" AssociatedControlID="uxExtensionId"
CssClass="span-4 last" runat="server" Text="* Extension Id:" />
        <ektronUI:TextField  ID="uxExtensionId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Get"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFileExtension" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();
            if (Regex.IsMatch(uxSiteId.Text, @"^\d+$"))
            {
                List<FileExtension> fileExtensionList =
aliasSettingsManager.GetAllExtensions(long.Parse(uxSiteId.Text));
                uxExtensionListView.Visible = true;
                uxExtensionListView.DataSource = fileExtensionList;
                uxExtensionListView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid SiteId.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetAllExtensionsOnly()

GetAllExtensionsOnly()

Retrieves list of FileExtension objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Returns

Returns a list of FileExtension objects.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Get"></ektronUI:Button>
    </li>
</ol>

<asp:ListView ID="uxExtensionListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >No records found for given input.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
```

```
                        FileExtension
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Container.DataItem %>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();

            List<string> fileExtensionList = aliasSettingsManager.GetAllExtensionsOnly
();
            uxExtensionListView.Visible = true;
            uxExtensionListView.DataSource = fileExtensionList;
            uxExtensionListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetAllExtensionsOnly (site ID)

```
GetAllExtensionsOnly(System.Int64)
```

Retrieves FileExtension objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Site ID

Parameters

- `siteId`. ID of site.

Returns

Returns FileExtension objects.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxSiteIdLabel" AssociatedControlID="uxSiteId"
CssClass="span-4 last" runat="server" Text="* Site Id:" />
        <ektronUI:TextField  ID="uxSiteId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Get"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxExtensionListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >No records found for given input.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        FileExtension
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
```

```
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Container.DataItem %>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Text.RegularExpressions;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();
            if (Regex.IsMatch(uxSiteId.Text, @"^\d+$"))
            {
                List<string> fileExtensionList =
aliasSettingsManager.GetAllExtensionsOnly(long.Parse(uxSiteId.Text));
                uxExtensionListView.Visible = true;
                uxExtensionListView.DataSource = fileExtensionList;
                uxExtensionListView.DataBind();
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid SiteId.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetExtension (ID)

```
GetExtension(System.Int64)
```

Retrieves a FileExtension object by extension ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Extension ID

Parameters

- `extensionId`. ID of extension.

Returns

Returns a FileExtension object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionIdLabel" AssociatedControlID="uxExtensionId"
CssClass="span-4 last" runat="server" Text="* Extension Id:" />
        <ektronUI:TextField  ID="uxExtensionId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Get"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFileExtension" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();
            long extensionId = 0;
            long.TryParse(uxExtensionId.Text, out extensionId);
            if (extensionId != 0)
            {
                FileExtension fileExtension = aliasSettingsManager.GetExtension
(extensionId);
                if (fileExtension != null && fileExtension.Id != 0)
                {

                    MessageUtilities.UpdateMessage(uxMessage, "Details for FileExtension
with Id " + fileExtension.Id.ToString() + " are below", Message.DisplayModes.Success);
                    uxFileExtension.Text = "FileExtension : " + fileExtension.Extension;
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
ExtensionId.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
ExtensionId.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update (aliasSetting)

```
Update(Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasSettings)
```

Updates the setting if the setting name and site ID matches in database; else inserts new record.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Site ID
- Default Replacement Character
- Is Alias Fallback Enabled
- Is Aliasing Enabled
- Is Folder Aliasing Enabled
- Is Group Aliasing Enabled
- Is Manual Aliasing Enabled
- Is RegEx Aliasing Enabled
- Is Taxonomy Aliasing Enabled
- Is URL Redirect Enabled
- Is User Aliasing Enabled
- Override Template
- Query String Action

Parameters

- `settings`. AliasSettings object.

Returns

Returns updated AliasSettings object.

Remarks

Validate the message item with `GetItem()` before you update the properties. Then, call `Update` with your modified UrlAliasData object.

NOTE: You cannot change all properties after the initial Add event, such as the UrlAliasAutoData .Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxSiteIdLabel" AssociatedControlID="uxSiteId"
CssClass="span-5 last" runat="server" Text="* Site Id:" />
        <ektronUI:TextField  ID="uxSiteId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDefaultReplacementCharacterLabel"
AssociatedControlID="uxDefaultReplacementCharacter" CssClass="span-5 last"
runat="server" Text=" DefaultReplacementCharacter:" />
        <ektronUI:TextField  ID="uxDefaultReplacementCharacter" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsAliasFallbackEnabledLabel"
AssociatedControlID="uxIsAliasFallbackEnabled" CssClass="span-5 last" runat="server"
Text=" IsAliasFallbackEnabled:" />
        <asp:DropDownList ID="uxIsAliasFallbackEnabled" runat="server">
```

```
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem Value="True">True</asp:ListItem>
            <asp:ListItem Value="False">False</asp:ListItem>
        </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxIsAliasingEnabledLabel"
AssociatedControlID="uxIsAliasingEnabled" CssClass="span-5 last" runat="server" Text="
IsAliasingEnabled:" />
        <asp:DropDownList ID="uxIsAliasingEnabled" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem Value="True">True</asp:ListItem>
            <asp:ListItem Value="False">False</asp:ListItem>
        </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxIsFolderAliasingEnabledLabel"
AssociatedControlID="uxIsFolderAliasingEnabled" CssClass="span-5 last" runat="server"
Text=" IsFolderAliasingEnabled:" />
        <asp:DropDownList ID="uxIsFolderAliasingEnabled" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem Value="True">True</asp:ListItem>
            <asp:ListItem Value="False">False</asp:ListItem>
        </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxIsGroupAliasingEnabledLabel"
AssociatedControlID="uxIsGroupAliasingEnabled" CssClass="span-5 last" runat="server"
Text=" IsGroupAliasingEnabled:" />
        <asp:DropDownList ID="uxIsGroupAliasingEnabled" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem Value="True">True</asp:ListItem>
            <asp:ListItem Value="False">False</asp:ListItem>
        </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxIsManualAliasingEnabledLabel"
AssociatedControlID="uxIsManualAliasingEnabled" CssClass="span-5 last" runat="server"
Text=" IsManualAliasingEnabled:" />
        <asp:DropDownList ID="uxIsManualAliasingEnabled" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem Value="True">True</asp:ListItem>
            <asp:ListItem Value="False">False</asp:ListItem>
        </asp:DropDownList>
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxIsRegExAliasingEnabledLabel"
AssociatedControlID="uxIsRegExAliasingEnabled" CssClass="span-5 last" runat="server"
Text=" IsRegExAliasingEnabled:" />
        <asp:DropDownList ID="uxIsRegExAliasingEnabled" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem Value="True">True</asp:ListItem>
            <asp:ListItem Value="False">False</asp:ListItem>
        </asp:DropDownList>
    </li>
        <li class="clearfix">
```

```
        <ektronUI:Label ID="uxIsTaxonomyAliasingEnabledLabel"
AssociatedControlID="uxIsTaxonomyAliasingEnabled" CssClass="span-5 last" runat="server"
Text=" IsTaxonomyAliasingEnabled:" />
        <asp:DropDownList ID="uxIsTaxonomyAliasingEnabled" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem Value="True">True</asp:ListItem>
            <asp:ListItem Value="False">False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsURLRedirectEnabledLabel"
AssociatedControlID="uxIsURLRedirectEnabled" CssClass="span-5 last" runat="server"
Text=" IsURLRedirectEnabled:" />
        <asp:DropDownList ID="uxIsURLRedirectEnabled" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem Value="True">True</asp:ListItem>
            <asp:ListItem Value="False">False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsUserAliasingEnabledLabel"
AssociatedControlID="uxIsUserAliasingEnabled" CssClass="span-5 last" runat="server"
Text=" IsUserAliasingEnabled:" />
        <asp:DropDownList ID="uxIsUserAliasingEnabled" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem Value="True">True</asp:ListItem>
            <asp:ListItem Value="False">False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxOverrideTemplateLabel"
AssociatedControlID="uxOverrideTemplate" CssClass="span-5 last" runat="server" Text="
OverrideTemplate:" />
        <asp:DropDownList ID="uxOverrideTemplate" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem Value="True">True</asp:ListItem>
            <asp:ListItem Value="False">False</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxQueryStringActionLabel"
AssociatedControlID="uxQueryStringAction" CssClass="span-5 last" runat="server" Text="
QueryStringAction:" />
        <asp:DropDownList ID="uxQueryStringAction" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem Value="0">None</asp:ListItem>
            <asp:ListItem Value="1">Replace</asp:ListItem>
            <asp:ListItem Value="2">Append</asp:ListItem>
            <asp:ListItem Value="3">Resolve</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-5" runat="server" Text="* -
Required" />
```

```
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using System.Text.RegularExpressions;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();
            string aliasSettingName = uxAliasSettingName.SelectedItem.Text;
            if (aliasSettingName != "Select" && uxAliasSettingValue.Text != "" &&
Regex.IsMatch(uxSiteId.Text, @"^\d+$"))
            {
                AliasSettingName settingName;
                if (aliasSettingName == "DefaultReplaceCharString")
                    settingName = AliasSettingName.DefaultReplaceCharString;
                else if (aliasSettingName == "DefaultReplacementCharacter")
                    settingName = AliasSettingName.DefaultReplacementCharacter;
                else if (aliasSettingName == "IsAliasFallbackEnabled")
                    settingName = AliasSettingName.IsAliasFallbackEnabled;
                else if (aliasSettingName == "IsAliasingEnabled")
                    settingName = AliasSettingName.IsAliasingEnabled;
                else if (aliasSettingName == "IsFolderAliasingEnabled")
                    settingName = AliasSettingName.IsFolderAliasingEnabled;
                else if (aliasSettingName == "IsGroupAliasingEnabled")
                    settingName = AliasSettingName.IsGroupAliasingEnabled;
                else if (aliasSettingName == "IsManualAliasingEnabled")
                    settingName = AliasSettingName.IsManualAliasingEnabled;
                else if (aliasSettingName == "IsRegExAliasingEnabled")
                    settingName = AliasSettingName.IsRegExAliasingEnabled;
                else if (aliasSettingName == "IsTaxonomyAliasingEnabled")
                    settingName = AliasSettingName.IsTaxonomyAliasingEnabled;
                else if (aliasSettingName == "IsURLRedirectEnabled")
                    settingName = AliasSettingName.IsURLRedirectEnabled;
                else if (aliasSettingName == "IsUserAliasingEnabled")
                    settingName = AliasSettingName.IsUserAliasingEnabled;
                else if (aliasSettingName == "OverrideTemplate")
                    settingName = AliasSettingName.OverrideTemplate;
                else
                    settingName = AliasSettingName.QueryStringAction;
                aliasSettingsManager.Update(settingName, uxAliasSettingValue.Text,
```

```
long.Parse(uxSiteId.Text));
                MessageUtilities.UpdateMessage(uxMessage, "The AliasSetting has been
Updated.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid values for
required fields.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update (name and ID)

```
Update
(Ektron.Cms.Settings.UrlAliasing.DataObjects.AliasSettingName,System.String,System.Int6
4)
```

Updates the setting if setting name and site ID matches in database; else inserts new record.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Setting
- * Alias Setting
- * Site ID

Parameters

- `settingName`. Name of the setting.
- `settingvalue`. Value of the setting.
- `siteID`. ID of the site.

Remarks

Validate the message item with `GetItem()` before you update the properties. Then, call `Update` with your modified UrlAliasData object.

NOTE: You cannot change all properties after the initial Add event, such as the UrlAliasAutoData .Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasSettingNameLabel"
AssociatedControlID="uxAliasSettingName" CssClass="span-3 last" runat="server" Text="*
Alias Setting Name:" />
        <asp:DropDownList ID="uxAliasSettingName" runat="server">
            <asp:ListItem>Select</asp:ListItem>
            <asp:ListItem>DefaultReplaceCharString</asp:ListItem>
            <asp:ListItem>DefaultReplacementCharacter</asp:ListItem>
            <asp:ListItem>IsAliasFallbackEnabled</asp:ListItem>
            <asp:ListItem>IsAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsFolderAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsGroupAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsManualAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsRegExAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsTaxonomyAliasingEnabled</asp:ListItem>
            <asp:ListItem>IsURLRedirectEnabled</asp:ListItem>
            <asp:ListItem>IsUserAliasingEnabled</asp:ListItem>
            <asp:ListItem>OverrideTemplate</asp:ListItem>
            <asp:ListItem>QueryStringAction</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasSettingValueLabel"
AssociatedControlID="uxAliasSettingValue" CssClass="span-3 last" runat="server" Text="*
Alias Setting Value:" />
        <ektronUI:TextField  ID="uxAliasSettingValue" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSiteIdLabel" AssociatedControlID="uxSiteId"
CssClass="span-3 last" runat="server" Text="* Site Id:" />
        <ektronUI:TextField  ID="uxSiteId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Text.RegularExpressions;
```

```
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AliasSettingsManager aliasSettingsManager = new AliasSettingsManager();
            if (Regex.IsMatch(uxSiteId.Text, @"^\d+$"))
            {
                AliasSettings settings = aliasSettingsManager.Get(long.Parse
(uxSiteId.Text));

                string defaultReplacementCharacter = uxDefaultReplacementCharacter.Text;
                if (defaultReplacementCharacter != string.Empty)
                {
                    if (defaultReplacementCharacter.Length != 1 || Regex.IsMatch
(defaultReplacementCharacter, @"[#*+[\]|?: <>'%.&]"))
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
DefaultReplacementCharacter.", Message.DisplayModes.Error);
                        uxPageMultiView.SetActiveView(uxViewMessage);
                        return;
                    }
                    else
                        settings.DefaultReplacementCharacter =
defaultReplacementCharacter;
                }

                if (uxIsAliasFallbackEnabled.SelectedValue != "Select")
                    settings.IsAliasFallbackEnabled = Convert.ToBoolean
(uxIsAliasFallbackEnabled.SelectedValue);

                if (uxIsAliasingEnabled.SelectedItem.Text != "Select")
                    settings.IsAliasingEnabled = Convert.ToBoolean
(uxIsAliasingEnabled.SelectedValue);

                if (uxIsFolderAliasingEnabled.SelectedValue != "Select")
                    settings.IsFolderAliasingEnabled = Convert.ToBoolean
(uxIsFolderAliasingEnabled.SelectedValue);

                if (uxIsGroupAliasingEnabled.SelectedValue != "Select")
                    settings.IsGroupAliasingEnabled = Convert.ToBoolean
(uxIsGroupAliasingEnabled.SelectedValue);

                if (uxIsManualAliasingEnabled.SelectedValue != "Select")
                    settings.IsManualAliasingEnabled = Convert.ToBoolean
(uxIsManualAliasingEnabled.SelectedValue);

                if (uxIsRegExAliasingEnabled.SelectedValue != "Select")
                    settings.IsRegExAliasingEnabled = Convert.ToBoolean
(uxIsRegExAliasingEnabled.SelectedValue);

                if (uxIsTaxonomyAliasingEnabled.SelectedValue != "Select")
```

```
                    settings.IsTaxonomyAliasingEnabled = Convert.ToBoolean
(uxIsTaxonomyAliasingEnabled.SelectedValue);

                if (uxIsURLRedirectEnabled.SelectedValue != "Select")
                    settings.IsURLRedirectEnabled = Convert.ToBoolean
(uxIsURLRedirectEnabled.SelectedValue);

                if (uxIsUserAliasingEnabled.SelectedValue != "Select")
                    settings.IsUserAliasingEnabled = Convert.ToBoolean
(uxIsUserAliasingEnabled.SelectedValue);

                if (uxOverrideTemplate.SelectedValue != "Select")
                    settings.OverrideTemplate = Convert.ToBoolean
(uxOverrideTemplate.SelectedValue);

                if (uxQueryStringAction.SelectedValue != "Select")
                    settings.QueryStringAction =
(Ektron.Cms.Common.EkEnumeration.QueryStringActionType)int.Parse
(uxQueryStringAction.SelectedValue);

                settings = aliasSettingsManager.Update(settings);
                MessageUtilities.UpdateMessage(uxMessage, "AliasSettings with SiteId " +
settings.SiteID.ToString() + " has been Updated.", Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid value for
SiteId.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

AliasSettings

Namespace

```
Ektron.Cms.Settings.UrlAliasing.DataObjects
```

Properties

- `DefaultReplaceCharList`

```
public System.Collections.Generic.List<ReplacementCharacter>
  DefaultReplaceCharList { set; get; }
```

- `DefaultReplaceCharString`

```
public string DefaultReplaceCharString { set; get; }
```

- DefaultReplacementCharacter

```
public string DefaultReplacementCharacter { set; get; }
```

- IsAliasFallbackEnabled

```
public bool IsAliasFallbackEnabled { set; get; }
```

- IsAliasingEnabled

```
public bool IsAliasingEnabled { set; get; }
```

- IsFolderAliasingEnabled

```
public bool IsFolderAliasingEnabled { set; get; }
```

- IsGroupAliasingEnabled

```
public bool IsGroupAliasingEnabled { set; get; }
```

- IsManualAliasingEnabled

```
public bool IsManualAliasingEnabled { set; get; }
```

- IsRegExAliasingEnabled

```
public bool IsRegExAliasingEnabled { set; get; }
```

- IsTaxonomyAliasingEnabled

```
public bool IsTaxonomyAliasingEnabled { set; get; }
```

- IsURLRedirectEnabled

```
public bool IsURLRedirectEnabled { set; get; }
```

- IsUserAliasingEnabled

```
public bool IsUserAliasingEnabled { set; get; }
```

- OverrideTemplate

```
public bool OverrideTemplate { set; get; }
```

- QueryStringAction

```
public EkEnumeration.QueryStringActionType
  QueryStringAction { set; get; }
```

- SiteID

```
public long SiteID { set; get; }
```

FileExtension

Namespace

```
Ektron.Cms.Settings.UrlAliasing.DataObjects
```

Properties

- Extension

```
public string Extension { set; get; }
```

- FileExtension.Equals (Ektron.Cms.Settings.UrlAliasing.DataObjects.FileExtension)

```
public bool
  Equals(Ektron.Cms.Settings.UrlAliasing.DataObjects.FileExtension other)
```

- Id

```
public long Id { set; get; }
```

- SiteId

```
public long SiteId { set; get; }
```

AutoAliasManager

8.50 and higher

The AutoAliasManager class manages automatic aliases, which lets you assign an alias to several content items at once. See Creating User-Friendly URLs with Aliasing in the Ektron Reference for information about aliases.

Namespace

```
Ektron.Cms.Framework.Settings.UrlAliasing.AutoAliasManager
```

Constructors

- `AutoAliasManager()`
- `AutoAliasManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *ClearCache* on page 1738
- *Delete* on page 1739
- *GetItem* on page 1741
- *GetList* on page 1743
- *GetTarget (alias and host)* on page 1745
- *GetTarget (alias, host and type)* on page 1747
- *Update* on page 1749

Add

```
Add(Ektron.Cms.Common.UrlAliasAutoData)
```

Adds a UrlAliasAutoData object.

The method populates the message.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Auto Alias Type
- * Extension
- Source ID
- Is Enabled

Parameters

- `data`. UrlAliasAutoData object.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAutoAliasTypeLabel" AssociatedControlID="uxAutoAliasType"
CssClass="span-3 last" runat="server" Text="* Auto Alias Type:" />
        <asp:DropDownList ID="uxAutoAliasType" runat="server">
            <asp:ListItem>Taxonomy</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionLabel" AssociatedControlID="uxExtension"
CssClass="span-3 last" runat="server" Text="* Extension:" />
        <asp:DropDownList ID="uxExtension" runat="server">
            <asp:ListItem>.aspx</asp:ListItem>
            <asp:ListItem>/</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSourceIdLabel" AssociatedControlID="uxSourceId"
CssClass="span-3 last" runat="server" Text=" Source Id:" />
        <ektronUI:TextField  ID="uxSourceId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="248"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-3 last" runat="server" Text=" Is Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEnabled" CssClass="span-
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AutoAliasManager autoAliasManager = new AutoAliasManager();
            UrlAliasAutoData urlAliasAutoData = new UrlAliasAutoData(-1,
EkEnumeration.AutoAliasType.Taxonomy, long.Parse(uxSourceId.Text) ,
uxExtension.SelectedItem.Text)
            {
                IsEnabled = uxIsEnabled.Checked,
                LanguageId = autoAliasManager.RequestInformation.ContentLanguage ,
                PageNameType = EkEnumeration.AutoAliasNameType.ContentId
            };

            autoAliasManager.Add(urlAliasAutoData);

            MessageUtilities.UpdateMessage(uxMessage, "Auto Alias added with Id " +
urlAliasAutoData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

ClearCache

```
ClearCache()
```

Clears the cache. This method has no parameters.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AutoAliasManager autoAliasManager = new AutoAliasManager();
            autoAliasManager.ClearCache();

            MessageUtilities.UpdateMessage(uxMessage, "Cache Cleared",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a UrlAliasAutoData object from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. Unique ID of the AutoAlias

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxAutoAliasIdLabel" AssociatedControlID="uxAutoAliasId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxAutoAliasId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
            </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long autoAliasId = long.Parse(uxAutoAliasId.Text);

            AutoAliasManager autoAliasManager = new AutoAliasManager();
            UrlAliasAutoData urlAliasAutoData = autoAliasManager.GetItem(autoAliasId);

            if (urlAliasAutoData != null)
            {
                autoAliasManager.Delete(autoAliasId);
                MessageUtilities.UpdateMessage(uxMessage,"Auto Alias deleted.",
Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
AutoAliasId.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.UrlAliasAutoData)
```

Retrieves the properties of a specific UrlAliasAutoData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. Unique ID of the AutoAlias.

Returns

UrlAliasAutoData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxManualAliasIdLabel" AssociatedControlID="uxManualAliasId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxManualAliasId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsEnabled" runat="server"></asp:Literal>
```

```
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSelectedPath" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAutoAliasType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFileExtension" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLanguageId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long autoAliasId = long.Parse(uxManualAliasId.Text);

            AutoAliasManager autoAliasManager = new AutoAliasManager();
            UrlAliasAutoData urlAliasAutoData = autoAliasManager.GetItem(autoAliasId);

            if (urlAliasAutoData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Details for auto alias with
ID " + urlAliasAutoData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxIsEnabled.Text = "Is Enabled : " + urlAliasAutoData.IsEnabled;
                uxSelectedPath.Text = "Selected Path : " +
urlAliasAutoData.SelectedPath;
                uxAutoAliasType.Text = "Auto Alias Type : " +
(EkEnumeration.AutoAliasType)urlAliasAutoData.AutoAliasType;
                uxFileExtension.Text = "File Extension : " +
urlAliasAutoData.FileExtension;
                uxLanguageId.Text = "Language Id : " + urlAliasAutoData.LanguageId ;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Auto Alias does not exists.",
Message.DisplayModes.Error);
            }
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(AutoAliasCriteria)
```

Retrieves lists of objects through the GetList(AutoAliasCriteria) method.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Manual Alias Property
- * Object Value

Parameters

- `criteria`. AutoAliasCriteria defining which AutoAlias to get.

Returns

List of UrlAliasAutoData object.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxManualAliasPropertyLabel"
AssociatedControlID="uxManualAliasProperty" CssClass="span-6 last" runat="server"
Text="Manual Alias Property:" />
        <asp:DropDownList ID="uxManualAliasProperty" runat="server">
            <asp:ListItem>AutoAliasType</asp:ListItem>
            <asp:ListItem>SelectedPath</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxAutoAliasDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Selected Path
                    </th>
                    <th>
                        Auto Alias Type
                    </th>
                    <th>
                        File Extension
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("SelectedPath")%>
            </td>
            <td class="devsite-method">
                <%# Eval("AutoAliasType")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FileExtension")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.UrlAliasing;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            AutoAliasCriteria criteria = new AutoAliasCriteria();
            if (uxManualAliasProperty.SelectedItem.Text == "AutoAliasType")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(AutoAliasProperty.AutoAliasType,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(AutoAliasProperty.SelectedPath,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            AutoAliasManager autoAliasManager = new AutoAliasManager();
            List<UrlAliasAutoData> urlAliasAutoDataList = autoAliasManager.GetList
(criteria);

            uxAutoAliasDataListView.Visible = true;
            uxAutoAliasDataListView.DataSource = urlAliasAutoDataList;
            uxAutoAliasDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTarget (alias and host)

```
GetTarget(System.String,System.String)
```

Retrieves the target data based on the URL alias and the host.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * URL Alias
- Host

Parameters

- `urlAlias`. urlAlias of the target to retrieve.
- `host`. Host of the target to retrieve.

Returns

Target string.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUrlAliasLabel" AssociatedControlID="uxUrlAlias"
CssClass="span-3 last" runat="server" Text="* Url Alias :" />
        <ektronUI:TextField ID="uxUrlAlias" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxHostLabel" AssociatedControlID="uxHost" CssClass="span-3
last" runat="server" Text=" Host :" />
        <ektronUI:TextField ID="uxHost" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Target"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AutoAliasManager autoAliasManager = new AutoAliasManager();
            string target = autoAliasManager.GetTarget(uxUrlAlias.Text, uxHost.Text);
            if (!string.IsNullOrEmpty(target))
            {
                MessageUtilities.UpdateMessage(uxMessage, "Target URL :     " + target,
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "There is No Target URL.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTarget (alias, host and type)

```
GetTarget(System.String,System.String,
Ektron.Cms.Common.EkEnumeration.AutoAliasType)
```

Retrieves the target data based on the URL alias, host, and AutoAliasType.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * URL Alias
- Host
- * Auto Alias Type

Parameters

- `urlAlias`. urlAlias of the Target to retrieve.
- `host`. Host of the target to retrieve.
- `autoAliasType`. AutoAliasType of the Target to retrieve.

Returns

Target string.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUrlAliasLabel" AssociatedControlID="uxUrlAlias"
CssClass="span-3 last" runat="server" Text="* Url Alias :" />
        <ektronUI:TextField ID="uxUrlAlias" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxHostLabel" AssociatedControlID="uxHost" CssClass="span-3
last" runat="server" Text=" Host :" />
        <ektronUI:TextField ID="uxHost" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAutoAliasTypeLabel" AssociatedControlID="uxAutoAliasType"
CssClass="span-3 last" runat="server" Text="* Auto Alias Type:" />
        <asp:DropDownList ID="uxAutoAliasType" runat="server">
            <asp:ListItem>Folder</asp:ListItem>
            <asp:ListItem>Taxonomy</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Target"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            AutoAliasManager autoAliasManager = new AutoAliasManager();
            EkEnumeration.AutoAliasType autoAliasType =
```

```
(uxAutoAliasType.SelectedItem.Text ==  "Folder" ? EkEnumeration.AutoAliasType.Folder :
EkEnumeration.AutoAliasType.Taxonomy );
            string target = autoAliasManager.GetTarget(uxUrlAlias.Text, uxHost.Text,
autoAliasType);

            MessageUtilities.UpdateMessage(uxMessage,  "Target URL :     " + target,
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.AutoAliasData)
```

Updates an existing UrlAutoAlias in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Extension
- Source ID
- * Is Enable

Parameters

- `data`. UrlAliasAutoData object

Returns

Returns the custom CmsData object updated.

Remarks

Validate the message item with `GetItem()` before you update the properties. Then, call `Update` with your modified UrlAliasAutoData object.

NOTE: You cannot change all properties after the initial Add event, such as the UrlAliasAutoData .Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAutoAliasIdLabel" AssociatedControlID="uxAutoAliasId"
CssClass="span-3 last" runat="server" Text="* Id:" />
        <ektronUI:TextField  ID="uxAutoAliasId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionLabel" AssociatedControlID="uxExtension"
CssClass="span-3 last" runat="server" Text="* Extension:" />
        <asp:DropDownList ID="uxExtension" runat="server">
            <asp:ListItem>.aspx</asp:ListItem>
            <asp:ListItem>/</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxSourceIdLabel" AssociatedControlID="uxSourceId"
CssClass="span-3 last" runat="server" Text=" Source Id:" />
        <ektronUI:TextField  ID="uxSourceId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-3 last" runat="server" Text="* Is Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEnabled" CssClass="span-
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            long autoAliasId = long.Parse(uxAutoAliasId.Text);

            AutoAliasManager autoAliasManager = new AutoAliasManager();
            UrlAliasAutoData urlAliasAutoData = autoAliasManager.GetItem(autoAliasId);

            if (urlAliasAutoData != null)
            {
                urlAliasAutoData.IsEnabled = uxIsEnabled.Checked;
                urlAliasAutoData.FileExtension = uxExtension.SelectedItem.Text;
                urlAliasAutoData.SourceId = (uxSourceId.Text != string.Empty ?
long.Parse(uxSourceId.Text) : urlAliasAutoData.SourceId);

                autoAliasManager.Update(urlAliasAutoData);

                MessageUtilities.UpdateMessage(uxMessage, "Auto Alias updated",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Auto Alias does not exists.",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

AutoAliasCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.UrlAliasing
```

Constructors

- `AutoAliasCriteria()`

  ```
  public AutoAliasCriteria()
  ```

- `AutoAliasCriteria(Ektron.Cms.UrlAliasing.AutoAliasProperty, EkEnumeration.OrderByDirection)`

  ```
  public AutoAliasCriteria(Ektron.Cms.UrlAliasing.AutoAliasProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `AutoAliasProperty` are:

- AutoAliasType
- Example
- ExcludedPath
- FileExtension
- IsEnabled
- LanguageId
- PageNameType
- ReplacementCharacter
- SelectedPath
- SiteId
- SourceId
- SourceName
- SourceParmName

AutoAliasType

Namespace

```
Ektron.Cms.Common.EkEnumeration
```

Properties

- `Folder`

```
public Const Folder As Ektron.Cms.Common.EkEnumeration.AutoAliasType = 2
```

- `Taxonomy`

```
public Const Taxonomy As Ektron.Cms.Common.EkEnumeration.AutoAliasType = 1
```

UrlAliasAutoData

Namespace

```
Ektron.Cms.Common
```

Properties

- `AliasName`. Gets or sets the alias name for the URL.

```
public string AliasName { set; get; }
```

- `AutoAliasType`. Gets or sets the automatic alias source type. Types are Taxonomy and Folder.

```
public EkEnumeration.AutoAliasType AutoAliasType { set; get; }
```

- `AutoId`. Gets the unique ID generated for this record.

```
public long AutoId { set; get; }
```

- `ContentId`

```
public long ContentId { set; get; }
```

- `Example`. Gets or sets a sample of the selected configuration.

```
public string Example { set; get; }
```

- `ExcludedPath`. Returns the folder or taxonomy path excluded from the selected path.

```
public string ExcludedPath { set; get; }
```

- `FileExtension`. Gets or sets the selected file extension.

```
public string FileExtension { set; get; }
```

- `FormattedSelectionPath`. Returns the folder or taxonomy path with modifications.

```
public string FormattedSelectedPath { get; }
```

- `HostName`. Gets the name of the host where the content resides.

```
public string HostName { set; get; }
```

- `Id`. Gets the unique ID generated for this record.

```
public override long Id { set; get; }
```

- `IsEnabled`. Gets or sets whether the configuration is active or not.

```
public bool IsEnabled { set; get; }
```

- `LanguageId`. Gets the language ID for the automatic alias.

```
public int LanguageId { set; get; }
```

- `PageNameType`. Gets or sets the name for the configuration. Available types are:
  - ContentTitle
  - ContentId
  - ContentIdAndLanguage

```
public EkEnumeration.AutoAliasNameType PageNameType { set; get; }
```

- `ReplacementCharacter`. Gets or sets the character that will replace special characters in the generated URL.

```
public string ReplacementCharacter { set; get; }
```

- `SelectedPath`. Returns the folder or taxonomy path selected by the user.

```
public string SelectedPath { set; get; }
```

- `SiteId`. Gets or sets the root folder ID of the site. If no site is specified, returns zero (0).

```
public long SiteId { set; get; }
```

- `SourceId`. Gets or sets either the taxonomy ID or the folder ID depending on the auto alias type selected.

```
public long SourceId { set; get; }
```

- `SourceName`. Gets the name of the taxonomy or folder selected depending on the auto alias type selected.

```
public string SourceName { set; get; }
```

- `SourceParmName`

```
public string SourceParmName { set; get; }
```

- `Target`. Gets or sets the destination for the alias.

```
public string Target { set; get; }
```

- `TimeStamp`

```
public System.DateTime TimeStamp { set; get; }
```

- `UrlAliasAutoData()`

```
public UrlAliasAutoData()
```

- `UrlAliasAutoData(long, EkEnumeration.AutoAliasType, long, string)`

```
public UrlAliasAutoData(long siteId,
  EkEnumeration.AutoAliasType autoAliasType,
  long sourceId, string fileExtension)
```

- `UrlAliasAutoData(long, EkEnumeration.AutoAliasType, long, string, long, string, string, int, string, bool, string, EkEnumeration.AutoAliasNameType, string, string, string, System.DateTime, long, string, string)`

```
public UrlAliasAutoData(long siteId,
  EkEnumeration.AutoAliasType autoAliasType,
  long sourceId, string fileExtension, long autoId,
  string aliasName, string target, int languageId,
  string sourceName, bool isEnabled, string example,
  EkEnumeration.AutoAliasNameType pagenametype,
  string replacementCharacter, string hostName,
  string selectedPath, System.DateTime timeStamp,
  long contentId, string excludedPath, string sourceParmname)
```

CommonAliasManager

8.50 and higher

The CommonAliasManager class gets alias and target data that is common to any alias.

Namespace

```
Ektron.Cms.Framework.Settings.UrlAliasing.CommonAliasManager
```

Constructors

- `CommonAliasManager()`
- `CommonAliasManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *GetContentAlias* below
- *GetTarget* on page 1757

GetContentAlias

```
GetContentAlias(System.Int64)
```

Retrieves the alias of a content item.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `contentId`. ID of the content.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxContentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Content Alias"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CommonAliasManager commonAliasManager = new CommonAliasManager();
            string contentAlias = commonAliasManager.GetContentAlias(long.Parse
(uxContentId.Text));

            MessageUtilities.UpdateMessage(uxMessage, "Content Alias :   " +
contentAlias, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTarget

```
GetTarget(System.Uri)
```

Retrieves the target of a uniform resource identifier (URI).

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * URI

Parameters

- `uri`

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUriLabel" AssociatedControlID="uxUri" CssClass="span-3
last" runat="server" Text="* Uri :" />
        <ektronUI:TextField ID="uxUri" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="http://[HostName]/OnTrek_Releases_
SelfServ_HelpDesk_Pro/" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Target"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
```

```
        {
            Uri uri = new Uri(uxUri.Text);

            CommonAliasManager commonAliasManager = new CommonAliasManager();

            string contentAlias = commonAliasManager.GetTarget(uri);

            MessageUtilities.UpdateMessage(uxMessage, "Target :   " + contentAlias,
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

CommunityAliasManager

8.50 and higher

The CommunityAliasManager class manages aliases that apply to communities.

Namespace

```
Ektron.Cms.Framework.Settings.UrlAliasing.CommunityAliasManager
```

Constructors

- `CommunityAliasManager()`
- `CommunityAliasManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Common.UrlAliasCommunityData)
```

Adds a UrlAliasCommunityData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Alias Type
- * Extension
- * Alias Path
- Is Enabled
- Primary

Parameters

- `data`

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityAliasTypeLabel"
AssociatedControlID="uxCommunityAliasType" CssClass="span-3 last" runat="server" Text="*
Community Alias Type:" />
        <asp:DropDownList ID="uxCommunityAliasType" runat="server">
            <asp:ListItem>User</asp:ListItem>
            <asp:ListItem>Group</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionLabel" AssociatedControlID="uxExtension"
CssClass="span-3 last" runat="server" Text="* Extension:" />
        <asp:DropDownList ID="uxExtension" runat="server">
            <asp:ListItem>.aspx</asp:ListItem>
            <asp:ListItem>/</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasPathLabel" AssociatedControlID="uxAliasPath"
CssClass="span-3 last" runat="server" Text="* Alias Path:" />
        <ektronUI:TextField  ID="uxAliasPath" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-3 last" runat="server" Text=" Is Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEnabled" CssClass="span-
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPrimaryLabel" AssociatedControlID="uxPrimary"
```

```
CssClass="span-3 last" runat="server" Text=" Primary:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxPrimary" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CommunityAliasManager communityAliasManager = new CommunityAliasManager();
            UrlAliasCommunityData urlAliasCommunityData = new UrlAliasCommunityData()
            {
                IsEnabled = uxIsEnabled.Checked,
                IsDefault = uxPrimary.Checked ,
                CommunityAliasType = (uxCommunityAliasType.SelectedItem.Value == "User"
? EkEnumeration.CommunityAliasType.Group : EkEnumeration.CommunityAliasType.User),
                FileExtension = uxExtension.SelectedItem.Value,
                AliasPath = uxAliasPath.Text,
                Example = uxAliasPath.Text + "/[" +
uxCommunityAliasType.SelectedItem.Value + "Name]" + uxExtension.SelectedItem.Value
            };

            communityAliasManager.Add(urlAliasCommunityData);

            MessageUtilities.UpdateMessage(uxMessage, "Community Alias added with Id " +
urlAliasCommunityData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes an UrlAliasCommunityData object from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. Community Alias ID.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityAliasIdLabel"
AssociatedControlID="uxCommunityAliasId" CssClass="span-4 last" runat="server" Text="*
Id:" />
        <ektronUI:TextField ID="uxCommunityAliasId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityAliasId = long.Parse(uxCommunityAliasId.Text);
            CommunityAliasManager communityAliasManager = new CommunityAliasManager();
            UrlAliasCommunityData urlAliasCommunityData = communityAliasManager.GetItem
(communityAliasId);
            if (urlAliasCommunityData != null)
            {
                communityAliasManager.Delete(communityAliasId);
                MessageUtilities.UpdateMessage(uxMessage, "Community Alias deleted.",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Activity",
Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetCommunityGroupAlias

```
GetCommunityGroupAlias(System.Int64)
```

Retrieves the alias of a community group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `groupId`. ID of the group.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxGroupIdLabel" AssociatedControlID="uxGroupId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Community Group Alias"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CommunityAliasManager communityAliasManager = new CommunityAliasManager();
            string communityGroupAlias = communityAliasManager.GetCommunityGroupAlias
(long.Parse(uxGroupId.Text));

            if(communityGroupAlias != string.Empty)
                 MessageUtilities.UpdateMessage(uxMessage, "community Group Alias :   "
+ communityGroupAlias, Message.DisplayModes.Success);
            else
             MessageUtilities.UpdateMessage(uxMessage, "community Group Alias does not
exists.", Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.UrlAliasCommunityData)
```

Retrieves the properties of a specific UrlAliasCommunityData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the UrlAliasCommunity to get.

Returns

UrlAliasCommunity details in a UrlAliasCommunityData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityAliasIdLabel"
AssociatedControlID="uxCommunityAliasId" CssClass="span-3 last" runat="server" Text="*
Id :" />
        <ektronUI:TextField ID="uxCommunityAliasId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAliasName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxCommunityAliasType" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFileExtension" runat="server"></asp:Literal>
```

```
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLanguageId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityAliasId = long.Parse(uxCommunityAliasId.Text);

            CommunityAliasManager communityAliasManager = new CommunityAliasManager();
            UrlAliasCommunityData urlAliasCommunityData = communityAliasManager.GetItem
(communityAliasId);

            if (urlAliasCommunityData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage,"Details for community alias
with ID " + urlAliasCommunityData.Id.ToString() + " are below",
Message.DisplayModes.Success);

                uxIsEnabled.Text = "Is Enabled : " + urlAliasCommunityData.IsEnabled;
                uxAliasName.Text = "Alias Name : " + urlAliasCommunityData.AliasName ;
                uxCommunityAliasType.Text = "Community Alias Type : " +
(EkEnumeration.CommunityAliasType)urlAliasCommunityData.CommunityAliasType;
                uxFileExtension.Text = "File Extension : " +
urlAliasCommunityData.FileExtension;
                uxLanguageId.Text = "Language Id : " + urlAliasCommunityData.LanguageId;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Auto Alias does not exists.",
Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
    }
```

GetList

```
GetList(Ektron.Cms.UrlAliasing.CommunityAliasCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Community Alias Property
- * Object Value

Parameters

- `criteria`. CommunityAliasCriteria used to specify, or filter, the list of UrlAliasCommunityData objects to return.

Returns

A list of UrlAliasCommunityData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

Remarks

Use ListView because it has a paging option and data grouping.

Use the default page size = 50.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityAliasPropertyLabel"
AssociatedControlID="uxCommunityAliasProperty" CssClass="span-6 last" runat="server"
Text="Community Alias Property:" />
        <asp:DropDownList ID="uxCommunityAliasProperty" runat="server">
            <asp:ListItem>CommunityAliasType</asp:ListItem>
            <asp:ListItem>AliasPath</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
```

```
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxUrlAliasCommunityDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        Alias Path
                    </th>
                    <th>
                        File Extension
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("AliasPath")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FileExtension")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Cms.Common;
using Ektron.Cms.UrlAliasing;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            CommunityAliasCriteria criteria = new CommunityAliasCriteria();
            if (uxCommunityAliasProperty.SelectedItem.Text == "CommunityAliasType")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CommunityAliasProperty.CommunityAliasType,
CriteriaFilterOperator.EqualTo , Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(CommunityAliasProperty.AliasPath,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            CommunityAliasManager communityAliasManager = new CommunityAliasManager();
            List<UrlAliasCommunityData> urlAliasCommunityDataList =
communityAliasManager.GetList(criteria);

            uxUrlAliasCommunityDataListView.Visible = true;
            uxUrlAliasCommunityDataListView.DataSource = urlAliasCommunityDataList;
            uxUrlAliasCommunityDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetUserAlias

```
GetUserAlias(System.Int64)
```

Retrieves the alias of a user.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
User Alias"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            CommunityAliasManager communityAliasManager = new CommunityAliasManager();
            string userAlias = communityAliasManager.GetUserAlias(long.Parse
(uxUserId.Text));
            if(userAlias != string.Empty)
                 MessageUtilities.UpdateMessage(uxMessage, "User Alias :    " +
userAlias, Message.DisplayModes.Success);
            else
             MessageUtilities.UpdateMessage(uxMessage, "User Alias does not exists. ",
Message.DisplayModes.Error);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

```
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Common.UrlAliasCommunityData)
```

Updates a UrlAliasCommunity, with details from the supplied UrlAliasCommunityData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Community Alias ID
- * Extension
- Alias Path
- * Is Enabled
- * Primary

Parameters

- `data`

Returns

Returns the custom CmsData object updated.

Remarks

Validate the community alias with `GetItem()` before you update the properties. Then, call `Update` with your modified UrlAliasCommunityData object.

NOTE: You cannot change all properties after the initial Add event, such as the UrlAliasCommunityData .Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxCommunityAliasIdLabel"
AssociatedControlID="uxCommunityAliasId" CssClass="span-4 last" runat="server" Text="*
Community Alias Id:" />
        <ektronUI:TextField  ID="uxCommunityAliasId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
```

```
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionLabel" AssociatedControlID="uxExtension"
CssClass="span-4 last" runat="server" Text="* Extension:" />
        <asp:DropDownList ID="uxExtension" runat="server">
            <asp:ListItem>.aspx</asp:ListItem>
            <asp:ListItem>/</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasPathLabel" AssociatedControlID="uxAliasPath"
CssClass="span-4 last" runat="server" Text=" Alias Path:" />
        <ektronUI:TextField  ID="uxAliasPath" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-4 last" runat="server" Text="* Is Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEnabled" CssClass="span-
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPrimaryLabel" AssociatedControlID="uxPrimary"
CssClass="span-4 last" runat="server" Text="* Primary:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxPrimary" CssClass="span-4"
runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>

    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long communityAliasId = long.Parse(uxCommunityAliasId.Text);
```

```
            CommunityAliasManager communityAliasManager = new CommunityAliasManager();
            UrlAliasCommunityData urlAliasCommunityData = communityAliasManager.GetItem
(communityAliasId);

            if (urlAliasCommunityData != null)
            {
                urlAliasCommunityData.IsEnabled = uxIsEnabled.Checked;
                urlAliasCommunityData.IsDefault = uxPrimary.Checked;
                urlAliasCommunityData.FileExtension = uxExtension.SelectedItem.Value;
                urlAliasCommunityData.AliasName = uxAliasPath.Text != string.Empty ?
uxAliasPath.Text : urlAliasCommunityData.AliasName;
                                urlAliasCommunityData.AliasPath = uxAliasPath.Text != string.Empty ?
uxAliasPath.Text : urlAliasCommunityData.AliasPath;
                urlAliasCommunityData.Example = uxAliasPath.Text != string.Empty ?
uxAliasPath.Text + "/[" + urlAliasCommunityData.CommunityAliasType.ToString() + "Name]"
+ uxExtension.SelectedItem.Value : urlAliasCommunityData.Example;

                communityAliasManager.Update(urlAliasCommunityData);

                MessageUtilities.UpdateMessage(uxMessage,  "Community Alias updated",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,"Community Alias does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

CommunityAliasCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.UrlAliasing
```

Constructors

- `CommunityAliasCriteria()`

  ```
  public CommunityAliasCriteria()
  ```

- `CommunityAliasCriteria (Ektron.Cms.UrlAliasing.CommunityAliasProperty,`

EkEnumeration.OrderByDirection)

```
public CommunityAliasCriteria(Ektron.Cms.UrlAliasing.CommunityAliasProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `CommunityAliasProperty` are:

- AliasPath
- CommunityAliasType
- Example
- FileExtension
- Id
- IsDefault
- IsEnabled
- LanguageId
- ReplacementCharacter
- SiteId
- SourceParmName

UrlAliasCommunityData

Namespace

```
Ektron.Cms.Common
```

Properties

- `AliasName`

  ```
  public string AliasName { set; get; }
  ```

- `AliasPath`

  ```
  public string AliasPath { set; get; }
  ```

- `CommunityAliasType`

  ```
  public EkEnumeration.CommunityAliasType
    CommunityAliasType { set; get; }
  ```

- `DisplayName`

  ```
  public string DisplayName { set; get; }
  ```

- `Example`. Gets or sets a sample of the selected configuration.

  ```
  public string Example { set; get; }
  ```

- `FileExtension`

  ```
  public string FileExtension { set; get; }
  ```

- `HostName`. Gets the name of the host where the content resides.

  ```
  public string HostName { set; get; }
  ```

- `Id`. Gets the unique ID generated for this record.

  ```
  public long Id { set; get; }
  ```

- `IsDefault`

```
public bool IsDefault { set; get; }
```

- `IsEnabled`. Gets or sets whether the configuration is active.

```
public bool IsEnabled { set; get; }
```

- `LanguageId`. Gets the language ID for the automatic alias.

```
public int LanguageId { set; get; }
```

- `ObjectId`

```
public long ObjectId { set; get; }
```

- `ReplacementCharacter`. Gets or sets the character that will replace special characters in the generated URL.

```
public string ReplacementCharacter { set; get; }
```

- `SiteId`

```
public long SiteId { set; get; }
```

- `SourceParmName`

```
public string SourceParmName { set; get; }
```

- `Target`. Gets the destination for the alias.

```
public string Target { set; get; }
```

- `TimeStamp`

```
public System.DateTime TimeStamp { set; get; }
```

- `UrlAliasCommunityData()`

```
public UrlAliasCommunityData()
```

- `UrlAliasCommunityData(long, EkEnumeration.CommunityAliasType, string, string)`

```
public UrlAliasCommunityData(long siteId,
  EkEnumeration.CommunityAliasType communityAliasType,
  string aliasPath, string fileExtension)
```

ManualAliasManager

8.50 and higher

The ManualAliasManager class manages manual aliases.

When using manual aliasing, you choose a page name and then specify the content to appear whenever anyone enters the URL using that name into a browser address field. For example, you can alias `www.example.com/cms/index.aspx` as `www.example.com/cms/Launch.html`. From then on, the content can be identified either by its original URL or its alias.

For information about manual aliasing, see Creating User-Friendly URLs with Aliasing in the Ektron Reference.

Namespace

```
Ektron.Cms.Framework.Settings.UrlAliasing.ManualAliasManager
```

Constructors

- `ManualAliasManager()`
- `ManualAliasManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* on the facing page
- *ClearCache* on page 1779
- *Delete* on page 1779
- *GetDefaultAlias* on page 1781
- *GetItem* on page 1783

- *GetList* on page 1785
- *Update* on page 1787

Add

```
Add(Ektron.Cms.Common.UrlAliasManualData)
```

Adds a URL alias based on information in a UrlAliasManualData object. The method populates the message.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Alias Name
- * Extension
- Is Enabled
- Is Default
- * Content ID

Parameters

- `data`. UrlAliasManualData data.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasNameLabel" AssociatedControlID="uxAliasName"
CssClass="span-3 last" runat="server" Text="* Alias Name:" />
        <ektronUI:TextField  ID="uxAliasName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionLabel" AssociatedControlID="uxExtension"
CssClass="span-3 last" runat="server" Text="* Extension:" />
        <asp:DropDownList ID="uxExtension" runat="server">
            <asp:ListItem>.aspx</asp:ListItem>
            <asp:ListItem>/</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-3 last" runat="server" Text=" Is Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEnabled" CssClass="span-
```

```
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsDefaultLabel" AssociatedControlID="uxIsDefault"
CssClass="span-3 last" runat="server" Text=" Is Default:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsDefault" CssClass="span-
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text="* Content Id:" />
        <ektronUI:TextField  ID="uxContentId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="30" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ManualAliasManager manualAliasManager = new ManualAliasManager();
            UrlAliasManualData urlAliasManualData = new UrlAliasManualData()
            {
                  IsEnabled = uxIsEnabled.Checked ,
                  IsDefault = uxIsDefault .Checked ,
                  AliasName = uxAliasName.Text ,
                  FileExtension = uxExtension.SelectedItem.Text,
                  ContentId = long.Parse(uxContentId.Text),
            };

            manualAliasManager.Add(urlAliasManualData);

            MessageUtilities.UpdateMessage(uxMessage, "Manual Alias added with Id " +
urlAliasManualData.Id, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

ClearCache

```
ClearCache()
```

Clears the cache. This method has no parameters.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            ManualAliasManager manualAliasManager = new ManualAliasManager();
            manualAliasManager.ClearCache();

            MessageUtilities.UpdateMessage(uxMessage, "Cache Cleared",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a URL alias (UrlAliasManualData) from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. Unique ID of the ManualAlias.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxManualAliasIdLabel" AssociatedControlID="uxManualAliasId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxManualAliasId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long manualAliasId = long.Parse(uxManualAliasId.Text);

            ManualAliasManager manualAliasManager = new ManualAliasManager();
            UrlAliasManualData urlAliasManualData = manualAliasManager.GetItem
(manualAliasId);
            if (urlAliasManualData != null)
            {
                manualAliasManager.Delete(manualAliasId);
                MessageUtilities.UpdateMessage(uxMessage,  "Manual Alias deleted.",
Message.DisplayModes.Success);
```

```
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Manual Alias does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetDefaultAlias

```
GetDefaultAlias(System.Int64)
```

Retrieves a content default alias.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. Content ID.

Returns

UrlAliasManualData.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxcontentIdLabel" AssociatedControlID="uxcontentId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxcontentId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Default Alias"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
```

```
<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAliasName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentIds" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDisplayAlias" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFileExtension" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long contentId = long.Parse(uxcontentId.Text);

            ManualAliasManager manualAliasManager = new ManualAliasManager();
            UrlAliasManualData urlAliasManualData = manualAliasManager.GetDefaultAlias
(contentId);

            if (urlAliasManualData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Details for default alias
for content ID " + contentId.ToString() + " are below", Message.DisplayModes.Success);

                uxAliasName.Text = "Alias Name : " + urlAliasManualData.AliasName;
                uxContentIds.Text = "Content Id : " + urlAliasManualData.ContentId;
                uxDisplayAlias.Text = "Display Alias : " +
urlAliasManualData.DisplayAlias;
                uxFileExtension.Text = "File Extension : " +
urlAliasManualData.FileExtension;
            }
            else
            {
```

```
                MessageUtilities.UpdateMessage(uxMessage, "Default Alias does not
exists.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, "Default Alias does not exists.",
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.UrlAliasManualData)
```

Retrieves the properties of a specific UrlAliasManualData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. Unique ID of the ManualAlias.

Returns

Locale data.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxManualAliasIdLabel" AssociatedControlID="uxManualAliasId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxManualAliasId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
```

```
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxAliasName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxContentId" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDisplayAlias" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFileExtension" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long manualAliasId = long.Parse(uxManualAliasId.Text);

            ManualAliasManager manualAliasManager = new ManualAliasManager();
            UrlAliasManualData urlAliasManualData = manualAliasManager.GetItem
(manualAliasId);

            if (urlAliasManualData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for manual alias with
ID " + urlAliasManualData.Id.ToString() + " are below", Message.DisplayModes.Success);

                uxAliasName.Text = "Alias Name : " + urlAliasManualData.AliasName ;
                uxContentId.Text = "Content Id : " + urlAliasManualData.ContentId;
                uxDisplayAlias.Text = "Display Alias : " +
urlAliasManualData.DisplayAlias;
                uxFileExtension.Text = "File Extension : " +
urlAliasManualData.FileExtension;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Manual Alias does not
exists.", Message.DisplayModes.Error);
```

```
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.UrlAliasing.ManualAliasCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- Manual Alias Property
- * Object Value

Parameters

- `criteria`. ManualAliasCriteria defining which UrlAliasManualData to get.

Returns

List of UrlAliasManualData data.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxManualAliasPropertyLabel"
AssociatedControlID="uxManualAliasProperty" CssClass="span-6 last" runat="server"
Text="Manual Alias Property:" />
        <asp:DropDownList ID="uxManualAliasProperty" runat="server">
            <asp:ListItem>Alias Id</asp:ListItem>
            <asp:ListItem>Alias Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
```

```
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxManualAliasDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Display Alias
                    </th>
                    <th>
                        File Extension
                    </th>
                    <th>
                        Content Id
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("DisplayAlias")%>
            </td>
            <td class="devsite-method">
                <%# Eval("FileExtension")%>
            </td>
            <td class="devsite-method">
                <%# Eval("ContentId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.UrlAliasing;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;

            ManualAliasCriteria criteria = new ManualAliasCriteria();
            if (uxManualAliasProperty.SelectedItem.Text == "Alias Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(ManualAliasProperty.AliasId,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(ManualAliasProperty.AliasName,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            ManualAliasManager manualAliasManager = new ManualAliasManager();
            List<UrlAliasManualData> urlAliasManualDataList = manualAliasManager.GetList
(criteria);

            uxManualAliasDataListView.Visible = true;
            uxManualAliasDataListView.DataSource = urlAliasManualDataList;
            uxManualAliasDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

Update(Ektron.Cms.UserCustomPropertyData)

Updates an existing URL alias based on information in a UrlAliasManualData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- Alias Name
- Extension
- Is Enabled
- Is Default
- Content ID

Parameters

- `data`. UrlAliasManualData data.

Returns

Returns the custom CmsData object added.

Remarks

Validate the URL alias with `GetItem()` before you update the properties. Then, call `Update` with your modified `UrlAliasManualData` object.

NOTE: You cannot change all properties after the initial `Add` event, such as the `UrlAliasManualData.Type`.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxManualAliasIdLabel" AssociatedControlID="uxManualAliasId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxManualAliasId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAliasNameLabel" AssociatedControlID="uxAliasName"
CssClass="span-3 last" runat="server" Text=" Alias Name:" />
        <ektronUI:TextField  ID="uxAliasName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExtensionLabel" AssociatedControlID="uxExtension"
CssClass="span-3 last" runat="server" Text=" Extension:" />
        <asp:DropDownList ID="uxExtension" runat="server">
            <asp:ListItem>SelectOne</asp:ListItem>
            <asp:ListItem>.aspx</asp:ListItem>
            <asp:ListItem>/</asp:ListItem>
        </asp:DropDownList>
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-3 last" runat="server" Text=" Is Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEnabled" CssClass="span-
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsDefaultLabel" AssociatedControlID="uxIsDefault"
CssClass="span-3 last" runat="server" Text=" Is Default:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsDefault" CssClass="span-
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxContentIdLabel" AssociatedControlID="uxContentId"
CssClass="span-3 last" runat="server" Text=" Content Id:" />
        <ektronUI:TextField  ID="uxContentId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long manualAliasId = long.Parse(uxManualAliasId.Text);

            ManualAliasManager manualAliasManager = new ManualAliasManager();
            UrlAliasManualData urlAliasManualData = manualAliasManager.GetItem
(manualAliasId);

            if (urlAliasManualData != null)
            {
                urlAliasManualData.IsEnabled = uxIsEnabled.Checked == true ?
uxIsEnabled.Checked : urlAliasManualData.IsEnabled;
                urlAliasManualData.IsDefault = uxIsDefault.Checked == true ?
```

```
uxIsDefault.Checked : urlAliasManualData.IsDefault;
                urlAliasManualData.AliasName = (uxAliasName.Text != string.Empty ?
uxAliasName.Text : urlAliasManualData.AliasName) ;
                urlAliasManualData.FileExtension = uxExtension.SelectedItem.Text !=
"SelectOne" ? uxExtension.SelectedItem.Text : urlAliasManualData.FileExtension;
                urlAliasManualData.ContentId = (uxContentId.Text != string.Empty ?
long.Parse(uxContentId.Text) :  urlAliasManualData.ContentId) ;

                manualAliasManager.Update(urlAliasManualData);
                MessageUtilities.UpdateMessage(uxMessage, "Manual Alias updated",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Manual Alias does not
exists.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

ManualAliasCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.UrlAliasing
```

Constructors

- `ManualAliasCriteria()`

  ```
  public ManualAliasCriteria()
  ```

- `ManualAliasCriteria(Ektron.Cms.UrlAliasing.ManualAliasProperty, EkEnumeration.OrderByDirection)`

  ```
  public ManualAliasCriteria(Ektron.Cms.UrlAliasing.ManualAliasProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `ManualAliasProperty` are:

  - AliasId
  - AliasName
  - AliasPageName
  - ContentId

- ContentLanguage
- ContentTitle
- DisplayAlias
- FileExtension
- HostName
- IsDefault
- LibraryId
- QueryString
- QueryStringAction
- SiteId
- Target

UrlAliasManualData

Namespace

```
Ektron.Cms.Common
```

Properties

- `AliasId`. Gets the unique ID generated for this record.
  ```
  public long AliasId { set; get; }
  ```
- `AliasName`. Gets or sets the alias name for the URL without the extension.
  ```
  public string AliasName { set; get; }
  ```
- `AliasPageName`. Gets the name of the alias without the query string.
  ```
  public string AliasPageName { set; get; }
  ```
- `ContentId`. Gets or sets the Content ID.
  ```
  public long ContentId { set; get; }
  ```
- `ContentLanguage`. Gets the content language.
  ```
  public int ContentLanguage { set; get; }
  ```
- `ContentTitle`. Gets the title of the content.
  ```
  public string ContentTitle { set; get; }
  ```
- `DisplayAlias`. Gets the alias URL with the extension and query parameters. This is a computed field.
  ```
  public string DisplayAlias { set; get; }
  ```
- `FileExtension`. Gets or sets the file extension for the alias.
  ```
  public string FileExtension { set; get; }
  ```
- `HostName`. Public string HostName { set; get; }
  ```
  public string HostName { set; get; }
  ```
- `Id`. Gets the unique ID generated for this record.
  ```
  public override long Id { set; get; }
  ```

- `IsDefault`. Gets or sets which alias name should show when the content has multiple aliases.

```
public bool IsDefault { set; get; }
```

- `IsEnabled`. Gets or sets whether the alias name is active.

```
public bool IsEnabled { set; get; }
```

- `LibraryId`. Gets or sets the Library ID for the content.

```
public long LibraryId { set; get; }
```

- `QueryString`. Gets or sets the querystring parameters to append to the aliased URL.

```
public string QueryString { set; get; }
```

- `QueryStringAction`. Gets or sets the querystringaction parameters.

```
public EkEnumeration.QueryStringActionType
  QueryStringAction { set; get; }
```

- `SiteId`. Gets the root folder ID of the site. If no site is specified, returns 0 (zero).

```
public long SiteId { set; get; }
```

- `Target`

```
public string Target { set; get; }
```

- `TimeStamp`. Gets or sets the destination URL for the alias.

```
public System.DateTime TimeStamp { set; get; }
```

- `UrlAliasManualData()`

```
public UrlAliasManualData()
```

- `UrlAliasManualData(long, long, string, string)`

```
public UrlAliasManualData(long contentId, long libraryId,
  string aliasName, string fileExtension)
```

- `UrlAliasManualData(long, string, string)`

```
public UrlAliasManualData(long contentId, string aliasName,
  string fileExtension)
```

RedirectManager

8.61 and higher

The RedirectManager class manages URL redirect addresses.

Namespace

```
Ektron.Cms.Framework.Settings.UrlAliasing.RedirectManager
```

Constructors

- `RedirectManager()`
- `RedirectManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

- *Add* below
- *Delete* on page 1796
- *DeleteAll* on page 1798
- *GetItem* on page 1799
- *GetList* on page 1801
- *GetTarget* on page 1804
- *Update* on page 1806

Add

```
Add(Ektron.Cms.Settings.UrlAliasing.DataObjects.RedirectData)
```

Adds a RedirectData object. RedirectData.Id is populated with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Site
- * Redirect Code
- Original URL
- New URL
- Active

Parameters

- `data`. RedirectData object.

Returns

Returns added RedirectData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxSiteLabel" AssociatedControlID="uxSite" CssClass="span-3
last" runat="server" Text="* Site:" />
        <asp:DropDownList ID="uxSite" runat="server">
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRedirectCodeLabel" AssociatedControlID="uxRedirectCode"
CssClass="span-3 last" runat="server" Text="* Redirect Code:" />
        <asp:DropDownList ID="uxRedirectCode" runat="server">
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxOriginalUrlLabel" AssociatedControlID="uxOriginalUrl"
CssClass="span-3 last" runat="server" Text=" Original Url:"/>
        <ektronUI:TextField  ID="uxOriginalUrl" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNewUrlLabel" AssociatedControlID="uxNewUrl"
CssClass="span-3 last" runat="server" Text=" New Url:" />
        <ektronUI:TextField  ID="uxNewUrl" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxActiveLabel" AssociatedControlID="uxActive"
CssClass="span-3 last" runat="server" Text=" Active:" />
        <asp:checkbox ID="uxActive" runat="server" Checked="true" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Net;
using Ektron.Cms.Common;
```

```
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RedirectManager redirectManager = new RedirectManager();
            Ektron.Cms.SiteAPI siteApi = new Ektron.Cms.SiteAPI();

            RedirectData redirectData = new RedirectData();
            redirectData.SiteId = long.Parse(uxSite.SelectedValue);
            redirectData.StatusCode = (HttpStatusCode)int.Parse
(uxRedirectCode.SelectedValue);
            string newUrl = uxNewUrl.Text;

            if (!string.IsNullOrWhiteSpace(uxOriginalUrl.Text))
            {
                string originalUrl = uxOriginalUrl.Text.TrimStart('/');
                if (originalUrl.Contains(" ") || originalUrl.Contains(":/"))
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
OriginalUrl.", Message.DisplayModes.Error);
                    uxPageMultiView.SetActiveView(uxViewMessage);
                    return;
                }

                if (originalUrl.ToLower().StartsWith
(siteApi.RequestInformationRef.ApplicationPath.Trim('/').ToLower()) ||
                    newUrl.TrimStart('/').ToLower().StartsWith
(siteApi.RequestInformationRef.ApplicationPath.Trim('/').ToLower()))
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Can not redirect to or
from the workarea.", Message.DisplayModes.Error);
                    uxPageMultiView.SetActiveView(uxViewMessage);
                    return;
                }

                RedirectCriteria criteria = new RedirectCriteria();
                criteria.AddFilter(RedirectProperty.SourceURL,
CriteriaFilterOperator.EqualTo, originalUrl);
                List<RedirectData> redirectList = redirectManager.GetList(criteria);
                if (redirectList != null && redirectList.Count > 0)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "There is already a
redirect associated with the original URL.", Message.DisplayModes.Error);
                    uxPageMultiView.SetActiveView(uxViewMessage);
                    return;
                }

                redirectData.SourceURL = originalUrl;
            }
            else
                redirectData.SourceURL = string.Empty;
```

```
            if (!string.IsNullOrWhiteSpace(newUrl))
            {
                if(newUrl.Contains(" "))
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
NewUrl.", Message.DisplayModes.Error);
                    uxPageMultiView.SetActiveView(uxViewMessage);
                    return;
                }

                if (!newUrl.Contains("://") && !newUrl.StartsWith("/"))
                    newUrl = "/" + newUrl;

                // if statuscode is 200 then NewUrl shoud be relative URL.
                if (HttpStatusCode.OK == (HttpStatusCode)int.Parse
(uxRedirectCode.SelectedValue) && !newUrl.StartsWith("/"))
                {
                    MessageUtilities.UpdateMessage(uxMessage, "New URL for Redirect Code
'200' should be an internal URL starting with '/'.", Message.DisplayModes.Error);
                    uxPageMultiView.SetActiveView(uxViewMessage);
                    return;
                }
                redirectData.TargetURL = newUrl;
            }
            else
            {
                if ((HttpStatusCode.NotFound == (HttpStatusCode)int.Parse
(uxRedirectCode.SelectedValue)))
                    redirectData.TargetURL = string.Empty;
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
NewUrl.", Message.DisplayModes.Error);
                    uxPageMultiView.SetActiveView(uxViewMessage);
                    return;
                }
            }

            redirectData.Active = uxActive.Checked;
            redirectData = redirectManager.Add(redirectData);
            MessageUtilities.UpdateMessage(uxMessage, "RedirectData is added with Id " +
redirectData.Id.ToString(), Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes redirect information from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Redirect ID

Parameters

- `Redirectid`. The ID of the redirect information.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRedirectIdLabel" AssociatedControlID="uxRedirectId"
CssClass="span-3 last" runat="server" Text="* Redirect Id:" />
        <ektronUI:TextField  ID="uxRedirectId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            RedirectManager redirectManager = new RedirectManager();
            long redirectId = 0;
            long.TryParse(uxRedirectId.Text, out redirectId);

            if (redirectId != 0)
            {
                RedirectData redirectData = redirectManager.GetItem(redirectId);
                if (redirectData != null && redirectData.Id != 0)
                {
                    redirectManager.Delete(redirectData.Id);
                    MessageUtilities.UpdateMessage(uxMessage, "RedirectData with Id " +
redirectData.Id.ToString() + " has been Deleted.", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
RedirectId.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
RedirectId.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

DeleteAll

```
Delete()
```

Deletes all redirect information.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete All"></ektronUI:Button>
    </li>
</ol>
```

```
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RedirectManager redirectManager = new RedirectManager();
            redirectManager.DeleteAll();
            MessageUtilities.UpdateMessage(uxMessage, "All Redirect Informations has
been Deleted.", Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(System.Int64)
```

Retrieves a RedirectData object by RedirectId.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Redirect ID

Parameters

- `Redirectid`. ID of the task category to retrieve.

Returns

Returns a RedirectData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRedirectIdLabel" AssociatedControlID="uxRedirectId"
CssClass="span-3 last" runat="server" Text="* Redirect Id:" />
        <ektronUI:TextField  ID="uxRedirectId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSite" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxRedirectCode" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxOriginalUrl" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxNewUrl" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxActive" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RedirectManager redirectManager = new RedirectManager();
```

```
            long redirectId = 0;
            long.TryParse(uxRedirectId.Text, out redirectId);

            if (redirectId != 0)
            {
                RedirectData redirectData = redirectManager.GetItem(redirectId);
                if (redirectData != null && redirectData.Id != 0)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Details for Redirect with
Id " + redirectData.Id.ToString() + " are below", Message.DisplayModes.Success);
                    uxSite.Text = "Site Id : " + redirectData.SiteId.ToString();
                    uxRedirectCode.Text = "Redirect Code : " +
redirectData.StatusCode.GetHashCode();
                    uxOriginalUrl.Text = "Original Url : " + redirectData.SourceURL;
                    uxNewUrl.Text = "New Url : " + redirectData.TargetURL;
                    uxActive.Text = "Active : " + redirectData.Active.ToString();
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
RedirectId.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
RedirectId.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.Settings.UrlAliasing.DataObjects.RedirectCriteria)
```

Retrieves list of RedirectData objects based on supplied criteria.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Object Value

Parameters

- `criteria`. RedirectCriteria used to retrieve the redirect information.

Returns

Returns a list of RedirectData objects.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRedirectPropertyLabel"
AssociatedControlID="uxRedirectProperty" CssClass="span-6 last" runat="server" Text="
Redirect Property:" />
        <asp:DropDownList ID="uxRedirectProperty" runat="server">
            <asp:ListItem>RedirectId</asp:ListItem>
            <asp:ListItem>OriginalUrl</asp:ListItem>
            <asp:ListItem>NewUrl</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:ListView ID="uxRedirectListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
    <EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Redirect Id
                    </th>
                    <th>
                        Original Url
                    </th>
                    <th>
                        New Url
                    </th>
                    <th>
                        Site Id
                    </th>
```

```
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("SourceURL")%>
            </td>
            <td class="devsite-method">
                <%# Eval("TargetURL")%>
            </td>
            <td class="devsite-method">
                <%# Eval("SiteId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RedirectManager redirectManager = new RedirectManager();
            string redirectProperty = uxRedirectProperty.SelectedValue;
            string objectValue = uxObjectValue.Text;

            RedirectCriteria criteria = new RedirectCriteria();
            criteria.OrderByField = RedirectProperty.Id;
            criteria.OrderByDirection =
Ektron.Cms.Common.EkEnumeration.OrderByDirection.Ascending;

            if (redirectProperty == "RedirectId")
                criteria.AddFilter(RedirectProperty.Id,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, objectValue);
            else if(redirectProperty == "OriginalUrl")
```

```
                criteria.AddFilter(RedirectProperty.SourceURL,
Ektron.Cms.Common.CriteriaFilterOperator.Contains, objectValue);
            else
                criteria.AddFilter(RedirectProperty.Target,
Ektron.Cms.Common.CriteriaFilterOperator.Contains, objectValue);

            List<RedirectData> redirectDataList = redirectManager.GetList(criteria);
            uxRedirectListView.Visible = true;
            uxRedirectListView.DataSource = redirectDataList;
            uxRedirectListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTarget

```
GetTarget(System.Uri)
```

Retrieves a RedirectData object by source URL.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Original URL

Parameters

- `url`. URL of source.

Returns

Returns a RedirectData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxOriginalUrlLabel" AssociatedControlID="uxOriginalUrl"
CssClass="span-3 last" runat="server" Text="* Original Url:" />
        <ektronUI:TextField  ID="uxOriginalUrl" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
```

```
Target"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSite" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxRedirectCode" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSourceUrl" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxNewUrl" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxActive" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RedirectManager redirectManager = new RedirectManager();
            string originalUrl = uxOriginalUrl.Text;
            if (!string.IsNullOrWhiteSpace(originalUrl))
            {
                Uri url = new Uri(Request.Url.Scheme + "://" + Request.Url.Authority +
"/" + originalUrl);
                RedirectData redirectData = redirectManager.GetTarget(url);
                if (redirectData != null && redirectData.Id != 0)
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Details for Redirect with
OriginalUrl " + originalUrl + " are below", Message.DisplayModes.Success);
                    uxSite.Text = "Site Id : " + redirectData.SiteId.ToString();
```

```
                    uxRedirectCode.Text = "Redirect Code : " +
redirectData.StatusCode.GetHashCode();
                    uxSourceUrl.Text = "Original Url : " + redirectData.SourceURL;
                    uxNewUrl.Text = "New Url : " + redirectData.TargetURL;
                    uxActive.Text = "Active : " + redirectData.Active.ToString();
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
OriginalUrl.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
OriginalUrl.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Settings.UrlAliasing.DataObjects.RedirectData)
```

Updates the supplied RedirectData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Redirect ID
- Redirect Code
- Original URL
- New URL
- Active

Parameters

- `data`. The RedirectData object to update.

Returns

Updated the RedirectData object.

Remarks

Validate the redirect item with `GetItem()` before you update the properties. Then, call `Update` with your modified RedirectData object.

NOTE: You cannot change all properties after the initial Add event, such as the RedirectData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRedirectIdLabel" AssociatedControlID="uxRedirectId"
CssClass="span-3 last" runat="server" Text="* Redirect Id:" />
        <ektronUI:TextField  ID="uxRedirectId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxRedirectCodeLabel" AssociatedControlID="uxRedirectCode"
CssClass="span-3 last" runat="server" Text=" Redirect Code:" />
        <asp:DropDownList ID="uxRedirectCode" runat="server">
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxOriginalUrlLabel" AssociatedControlID="uxOriginalUrl"
CssClass="span-3 last" runat="server" Text=" Original Url:"/>
        <ektronUI:TextField  ID="uxOriginalUrl" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text="" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxNewUrlLabel" AssociatedControlID="uxNewUrl"
CssClass="span-3 last" runat="server" Text=" New Url:" />
        <ektronUI:TextField  ID="uxNewUrl" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"  Text=""  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxActiveLabel" AssociatedControlID="uxActive"
CssClass="span-3 last" runat="server" Text=" Active:" />
        <asp:checkbox ID="uxActive" runat="server" Checked="true" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Net;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RedirectManager redirectManager = new RedirectManager();
            Ektron.Cms.SiteAPI siteApi = new Ektron.Cms.SiteAPI();
            RedirectData redirectData = new RedirectData();

            long redirectId = 0;
            long.TryParse(uxRedirectId.Text, out redirectId);

            if (redirectId != 0)
            {
                redirectData = redirectManager.GetItem(redirectId);
                if (redirectData != null && redirectData.Id != 0)
                {
                    string newUrl = uxNewUrl.Text;
                    if (uxRedirectCode.SelectedValue != "Select" &&
redirectData.StatusCode != (HttpStatusCode)int.Parse(uxRedirectCode.SelectedValue))
                    {
                        if (HttpStatusCode.OK == (HttpStatusCode)int.Parse
(uxRedirectCode.SelectedValue))
                        {
                            if (!string.IsNullOrWhiteSpace(newUrl))
                            {
                                if (newUrl.Contains("://"))
                                {
                                    MessageUtilities.UpdateMessage(uxMessage, "New URL
for Redirect Code '200' should be an internal URL starting with '/'.",
Message.DisplayModes.Error);
                                    uxPageMultiView.SetActiveView(uxViewMessage);
                                    return;
                                }
                            }
                            else
                            {
                                if (redirectData.TargetURL.Contains("://"))
                                {
                                    MessageUtilities.UpdateMessage(uxMessage, "New URL
for Redirect Code '200' should be an internal URL starting with '/'.",
Message.DisplayModes.Error);
                                    uxPageMultiView.SetActiveView(uxViewMessage);
                                    return;
                                }
                            }
                        }
```

```
                        if (HttpStatusCode.NotFound != (HttpStatusCode)int.Parse
(uxRedirectCode.SelectedValue) &&
                            string.IsNullOrWhiteSpace(redirectData.TargetURL) &&
string.IsNullOrWhiteSpace(newUrl))
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid NewUrl.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
                            return;
                        }

                        redirectData.StatusCode = (HttpStatusCode)int.Parse
(uxRedirectCode.SelectedValue);
                    }


                    if (!string.IsNullOrWhiteSpace(uxOriginalUrl.Text) &&
uxOriginalUrl.Text.TrimStart('/').ToLower() != redirectData.SourceURL.TrimStart
('/').ToLower())
                    {
                        string originalUrl = uxOriginalUrl.Text.TrimStart('/');
                        if (originalUrl.Contains(" ") || originalUrl.Contains(":/"))
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid OriginalUrl.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
                            return;
                        }

                        if (originalUrl.ToLower().StartsWith
(siteApi.RequestInformationRef.ApplicationPath.Trim('/').ToLower()) ||
                            newUrl.TrimStart('/').ToLower().StartsWith
(siteApi.RequestInformationRef.ApplicationPath.Trim('/').ToLower()))
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "Can not redirect
to or from the workarea.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
                            return;
                        }

                        RedirectCriteria criteria = new RedirectCriteria();
                        criteria.AddFilter(RedirectProperty.SourceURL,
CriteriaFilterOperator.EqualTo, originalUrl);
                        List<RedirectData> redirectList = redirectManager.GetList
(criteria);
                        if (redirectList != null && redirectList.Count > 0)
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "There is already
a redirect associated with the original URL.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
                            return;
                        }

                        redirectData.SourceURL = originalUrl;
                    }
```

```
                    if (!string.IsNullOrWhiteSpace(newUrl) && newUrl.TrimStart
('/').ToLower() != redirectData.TargetURL.TrimStart('/').ToLower())
                    {
                        if (newUrl.Contains(" "))
                        {
                            MessageUtilities.UpdateMessage(uxMessage, "Please enter
valid NewUrl.", Message.DisplayModes.Error);
                            uxPageMultiView.SetActiveView(uxViewMessage);
                            return;
                        }

                        if (!newUrl.Contains("://") && !newUrl.StartsWith("/"))
                            newUrl = "/" + newUrl;

                        redirectData.TargetURL = newUrl;
                    }

                    redirectData.Active = uxActive.Checked;
                    redirectData = redirectManager.Update(redirectData);
                    MessageUtilities.UpdateMessage(uxMessage, "RedirectData is updated
with Id " + redirectData.Id.ToString(), Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
RedirectId.", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please enter valid
RedirectId.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

RedirectCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.Settings.UrlAliasing.DataObjects
```

Constructors

- `RedirectCriteria()`

```
public RedirectCriteria()
```

- `RedirectCriteria (Ektron.Cms.Settings.UrlAliasing.DataObjects.RedirectProperty, EkEnumeration.OrderByDirection)`

```
public RedirectCriteria
(Ektron.Cms.Settings.UrlAliasing.DataObjects.RedirectProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `RedirectProperty` are:

- Active
- Id
- SiteId
- SourceURL
- StatusCode
- Target

RedirectData

Namespace

```
Ektron.Cms.Settings.UrlAliasing.DataObjects
```

Properties

- `Active`

```
public bool Active { set; get; }
```

- `Id`

```
public long Id { set; get; }
```

- `SiteId`

```
public long SiteId { set; get; }
```

- `SourceURL`

```
public string SourceURL { set; get; }
```

- `StatusCode`

```
public System.Net.HttpStatusCode StatusCode { set; get; }
```

- `TargetURL`

```
public string TargetURL { set; get; }
```

RegExAliasManager

8.50 and higher

The RegExAliasManager class manages regular expressions that replace URLs that cannot be read. See *Creating RegEx Expressions* in Creating User-Friendly URLs with Aliasing in the Ektron Reference for information about RegEx expressions.

Namespace

```
Ektron.Cms.Framework.Settings.UrlAliasing.RegExAliasManager
```

Constructors

- `RegExAliasManager()`
- `RegExAliasManager(ApiAccessMode)`

Properties

- `ApiMode`. Gets or sets the current API access mode. If set to `Admin`, the API runs with the permissions of an administrator.
- `ApplicationPath`. 9.00 and higher Gets the application path to the Workarea.
- `ContentLanguage`. Gets or sets the current content language.
- `InPreviewMode`. Gets or sets the preview mode and returns true if the site is in preview mode.
- `IsCommerceEnabled`. 8.70 and higher Checks for a commerce license.
- `RegExAliasManagerService`. Returns an instance of the business objects manager service.
- `RequestInformation`. Gets information about the current request.
- `SitePath`. Gets the site path.
- `UserId`. Gets or sets the ID of the currently logged-in user.

Methods

Add

```
Add(Ektron.Cms.Common.UrlAliasRegExData)
```

Adds an item based on information in an UrlAliasRegExData object. The method populates the message.Id with the newly created ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Expression Name
- * Expression
- * Expression Map
- * Example
- Is Enabled
- Priority

Parameters

- `data`. The UrlAliasRegExData object to add.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxExpressionNameLabel"
AssociatedControlID="uxExpressionName" CssClass="span-4 last" runat="server" Text="*
Expression Name:" />
        <ektronUI:TextField  ID="uxExpressionName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExpressionLabel" AssociatedControlID="uxExpression"
CssClass="span-4 last" runat="server" Text="* Expression:" />
        <ektronUI:TextField  ID="uxExpression" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="(\d{4})/(\d{2})/(\d{2})/Default.aspx" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExpressionMapLabel" AssociatedControlID="uxExpressionMap"
CssClass="span-4 last" runat="server" Text="* Expression Map:" />
        <ektronUI:TextField  ID="uxExpressionMap" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="PageName.aspx?year=$1&month=$2&day=$3"
/>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExampleLabel" AssociatedControlID="uxExample"
CssClass="span-4 last" runat="server" Text="* Example:" />
```

```
        <ektronUI:TextField  ID="uxExample" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="2000/02/02/Default.aspx" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-4 last" runat="server" Text=" Is Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEnabled" CssClass="span-
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPriorityLabel" AssociatedControlID="uxPriority"
CssClass="span-4 last" runat="server" Text=" Priority:" />
        <asp:DropDownList ID="uxPriority" runat="server">
            <asp:ListItem Value="100">High</asp:ListItem>
            <asp:ListItem Value="200">Low</asp:ListItem>
            <asp:ListItem Value="300">Medium</asp:ListItem>
            <asp:ListItem Value="10000">None</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
using Ektron.Site.Developer;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long siteId = 0;

            RegExAliasManager regExAliasManager = new RegExAliasManager();
            UrlAliasRegExData urlAliasRegExData = new UrlAliasRegExData
(uxExpressionName.Text, siteId, uxExpression.Text,  uxExpressionMap.Text)
                {
                    Priority = (EkEnumeration.RegExPriority)int.Parse
(uxPriority.SelectedItem.Value),
                    TransformedUrl = uxExample.Text ,
                    IsEnabled = uxIsEnabled.Checked
                };
```

```
            regExAliasManager.Add(urlAliasRegExData);

            MessageUtilities.UpdateMessage(uxMessage,  "Regex Alias added with Id " +
urlAliasRegExData.RegExId, Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

ClearCache

```
ClearCache()
```

Clears the cache. This method has no parameters.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Clear Cache"></ektronUI:Button>
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RegExAliasManager regExAliasManager = new RegExAliasManager();
            regExAliasManager.ClearCache();
```

```
            MessageUtilities.UpdateMessage(uxMessage, "Cache Cleared",
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Delete

```
Delete(System.Int64)
```

Deletes an item (UrlAliasRegExData) from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the UrlAliasRegEx to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRegExAliasIdLabel" AssociatedControlID="uxRegExAliasId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField ID="uxRegExAliasId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
                <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
                    Text="Status:" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
```

```
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
using Ektron.Site.Developer;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long regExAliasId = long.Parse(uxRegExAliasId.Text);

            RegExAliasManager regExAliasManager = new RegExAliasManager();
            UrlAliasRegExData urlAliasRegExData = regExAliasManager.GetItem
(regExAliasId);

            if (urlAliasRegExData != null)
            {
                regExAliasManager.Delete(regExAliasId);
                MessageUtilities.UpdateMessage(uxMessage, "RegEx Alias deleted.",
Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid RegEx Alias
ID.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetItem

```
GetItem(Ektron.Cms.UrlAliasRegExData)
```

Retrieves the properties of a specific UrlAliasRegExData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID

Parameters

- `id`. ID of the UrlAliasRegEx to retrieve.

Returns

UrlAliasRegExData.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRegExAliasIdLabel" AssociatedControlID="uxRegExAliasId"
CssClass="span-3 last" runat="server" Text="* Id :" />
        <ektronUI:TextField ID="uxRegExAliasId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Item"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxExpressionName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxExpressionMap" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxIsEnabled" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxExpression" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxSiteId" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
using Ektron.Site.Developer;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long regExAliasId = long.Parse(uxRegExAliasId.Text);

            RegExAliasManager regExAliasManager = new RegExAliasManager();
            UrlAliasRegExData urlAliasRegExData = regExAliasManager.GetItem
(regExAliasId);

            if (urlAliasRegExData != null)
            {
                MessageUtilities.UpdateMessage(uxMessage, "Details for RegEx alias with
ID " + urlAliasRegExData.RegExId.ToString() + " are below",
Message.DisplayModes.Success);

                uxExpressionName.Text = "Expression Name : " +
urlAliasRegExData.ExpressionName;
                uxExpressionMap.Text = "Expression Map : " +
urlAliasRegExData.ExpressionMap;
                uxExpression.Text = "File Extension : " + urlAliasRegExData.Expression;
                uxIsEnabled.Text = "Is Enabled : " + urlAliasRegExData.IsEnabled;
                uxSiteId.Text = "Site Id : " + urlAliasRegExData.SiteId ;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "RegEx Alias does not
exists.", Message.DisplayModes.Error);
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetList

```
GetList(Ektron.Cms.UrlAliasing.RegExAliasCriteria)
```

Retrieves a list of UrlAliasRegExData objects based upon the supplied criteria.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- RegEx Alias Property
- * Object Value

Parameters

- `criteria`. RegExAliasCriteria used to retrieve UrlAliasRegExData.

Returns

A list of UrlAliasRegExData.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRegExAliasPropertyLabel"
AssociatedControlID="uxRegExAliasProperty" CssClass="span-6 last" runat="server"
Text="RegEx Alias Property:" />
        <asp:DropDownList ID="uxRegExAliasProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>ExpressionName</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last" runat="server" Text="* ObjectValue:" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
List"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-6" runat="server" Text="* -
Required" />
    </li>
</ol>

<ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
<asp:ListView ID="urlAliasRegExDataListView" runat="server"
ItemPlaceholderID="aspItemPlaceholder" Visible="false">
<EmptyDataTemplate >Criteria did not match any records.</EmptyDataTemplate>
<LayoutTemplate>
    <table class="devsite-api-method">
```

```
        <thead>
            <tr>
                <th>
                    RegExId
                </th>
                <th>
                    Expression
                </th>
                <th>
                    ExpressionName
                </th>
            </tr>
        </thead>
        <tbody>
            <asp:PlaceHolder ID="aspItemPlaceholder" runat="server"></asp:PlaceHolder>
        </tbody>
    </table>
</LayoutTemplate>
<ItemTemplate>
    <tr>
        <td class="devsite-method">
            <%# Eval("RegExId")%>
        </td>
        <td class="devsite-method">
            <%# Eval("Expression")%>
        </td>
        <td class="devsite-method">
            <%# Eval("ExpressionName")%>
        </td>
    </tr>
</ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
using Ektron.Site.Developer;
using Ektron.Cms.UrlAliasing;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RegExAliasManager regExAliasManager = new RegExAliasManager();
            RegExAliasCriteria criteria = new RegExAliasCriteria();
            object Objectvalue;

            if (uxRegExAliasProperty.SelectedItem.Text == "Id")
```

```
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(RegExAliasProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(RegExAliasProperty.ExpressionName,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            ManualAliasManager manualAliasManager = new ManualAliasManager();
            List<UrlAliasRegExData> urlAliasRegExDataList = regExAliasManager.GetList
(criteria);

            urlAliasRegExDataListView.Visible = true;
            urlAliasRegExDataListView.DataSource = urlAliasRegExDataList;
            urlAliasRegExDataListView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

GetTarget

```
GetTarget(System.String,System.String)
```

Retrieves the target of a uniform resource locator (URL) alias.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * URL Alias
- Host

Parameters

- `urlAlias`. Alias.
- `host`. Host.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUrlAliasLabel" AssociatedControlID="uxUrlAlias"
CssClass="span-3 last" runat="server" Text="* Url Alias :" />
        <ektronUI:TextField ID="uxUrlAlias" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="CEOBlog/" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxHostLabel" AssociatedControlID="uxHost" CssClass="span-3
last" runat="server" Text=" Host :" />
        <ektronUI:TextField ID="uxHost" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click" Text="Get
Target"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-3" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.Settings.UrlAliasing;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            RegExAliasManager regExAliasManager = new RegExAliasManager();
            string target = regExAliasManager.GetTarget(uxUrlAlias.Text, uxHost.Text);

            MessageUtilities.UpdateMessage(uxMessage, "Target URL :     " + target,
Message.DisplayModes.Success);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Update

```
Update(Ektron.Cms.Common.UrlAliasRegExData)
```

Updates an existing item based on information in an UrlAliasRegExData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * ID
- * Expression Name
- * Expression
- * Expression Map
- * Example
- Is Enabled
- Priority

Parameters

- `data`. The UrlAliasRegExData object to update.

Returns

Returns the custom CmsData object updated.

Remarks

Validate the message with `GetItem()` before you update the properties. Then, call `Update` with your modified UrlAliasRegExData object.

NOTE: You cannot change all properties after the initial Add event, such as the UrlAliasRegExData .Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxRegExAliasIdLabel" AssociatedControlID="uxRegExAliasId"
CssClass="span-4 last" runat="server" Text="* Id:" />
        <ektronUI:TextField  ID="uxRegExAliasId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="5" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExpressionNameLabel"
AssociatedControlID="uxExpressionName" CssClass="span-4 last" runat="server" Text="*
Expression Name:" />
        <ektronUI:TextField  ID="uxExpressionName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="CEO Blog" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxExpressionLabel" AssociatedControlID="uxExpression"
CssClass="span-4 last" runat="server" Text="* Expression:" />
        <ektronUI:TextField  ID="uxExpression" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="CEOBlog/?$" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExpressionMapLabel" AssociatedControlID="uxExpressionMap"
CssClass="span-4 last" runat="server" Text="* Expression Map:" />
        <ektronUI:TextField  ID="uxExpressionMap" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
Text="community.blog.aspx?blogid=149&SelTaxonomyID=261" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxExampleLabel" AssociatedControlID="uxExample"
CssClass="span-4 last" runat="server" Text="* Example:" />
        <ektronUI:TextField  ID="uxExample" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="CEOBlog/" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIsEnabledLabel" AssociatedControlID="uxIsEnabled"
CssClass="span-4 last" runat="server" Text=" Is Enabled:" />
        <ektronUI:Button DisplayMode="Checkbox" Text="" ID="uxIsEnabled" CssClass="span-
4" runat="server" ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPriorityLabel" AssociatedControlID="uxPriority"
CssClass="span-4 last" runat="server" Text=" Priority:" />
        <asp:DropDownList ID="uxPriority" runat="server">
            <asp:ListItem Value="100">High</asp:ListItem>
            <asp:ListItem Value="200">Low</asp:ListItem>
            <asp:ListItem Value="300">Medium</asp:ListItem>
            <asp:ListItem Value="10000">None</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Button id="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Cms.Common;
using Ektron.Cms.Framework.Settings.UrlAliasing;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Settings.UrlAliasing.DataObjects;
using Ektron.Site.Developer;
using Ektron.Cms;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            long regExAliasId = long.Parse(uxRegExAliasId.Text);

            RegExAliasManager regExAliasManager = new RegExAliasManager();
            UrlAliasRegExData urlAliasRegExData = regExAliasManager.GetItem
(regExAliasId);

            if (urlAliasRegExData != null)
            {
                urlAliasRegExData.Expression = (uxExpression.Text != string.Empty ?
uxExpression.Text : urlAliasRegExData.Expression);
                urlAliasRegExData.ExpressionName = (uxExpressionName.Text !=
string.Empty ? uxExpressionName.Text : urlAliasRegExData.ExpressionName);
                urlAliasRegExData.ExpressionMap = (uxExpressionMap.Text != string.Empty
? uxExpressionMap.Text : urlAliasRegExData.ExpressionMap);
                urlAliasRegExData.Priority = (EkEnumeration.RegExPriority)int.Parse
(uxPriority.SelectedItem.Value);
                urlAliasRegExData.TransformedUrl = (uxExample.Text != string.Empty ?
uxExample.Text : urlAliasRegExData.TransformedUrl);
                urlAliasRegExData.IsEnabled = uxIsEnabled.Checked;

                regExAliasManager.Update(urlAliasRegExData);

                MessageUtilities.UpdateMessage(uxMessage, "Regex Alias updated",
Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "RegEx Alias does not
exists.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
    }
```

Data Classes

RegExAliasCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.UrlAliasing
```

Constructors

- `RegExAliasCriteria()`

```
public RegExAliasCriteria()
```

- `RegExAliasCriteria(Ektron.Cms.UrlAliasing.RegExAliasProperty, EkEnumeration.OrderByDirection)`

```
public RegExAliasCriteria(Ektron.Cms.UrlAliasing.RegExAliasProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `RegExAliasProperty` are:

- Expression
- ExpressionMap
- ExpressionName
- Id
- IsEnabled
- SiteId
- SortOrder
- TransformedUrl

UrlAliasRegExData

Namespace

```
Ektron.Cms
```

Properties

- `Expression`. Gets or sets the regular expression to be used.

```
public string Expression { set; get; }
```

- `ExpressionMap`. Gets or sets the destination for the expression.

```
public string ExpressionMap { set; get; }
```

- `ExpressionName`. Gets or sets a descriptive name for the expression.

```
public string ExpressionName { set; get; }
```

- `IsEnabled`. Gets or sets whether the expression is active.

```
public bool IsEnabled { set; get; }
```

- `Priority`. Gets or sets the look up order for the expression list. Priorities are:
  - None
  - High
  - Medium
  - Low

```
public EkEnumeration.RegExPriority Priority { set; get; }
```

- `RegExId`. Gets the unique ID generated for each record.

```
public long RegExId { set; get; }
```

- `SiteId`. Gets or sets the root folder ID of the site. Returns zero if no site is specified.

```
public long SiteId { set; get; }
```

- `TimeStamp`

```
public System.DateTime TimeStamp { set; get; }
```

- `TransformedUrl`. Gets or sets a sample of how the ExpressionMap URL will look.

```
public string TransformedUrl { set; get; }
```

- `UrlAliasRegExData()`

```
public UrlAliasRegExData()
```

- `UrlAliasRegExData(string, long, string, string)`

```
public UrlAliasRegExData(string expressionName,
  long siteId, string expression, string expressionMap)
```

- `UrlAliasRegExData(string, long, string, string, long, string, bool, EkEnumeration.RegExPriority, System.DateTime)`

```
public UrlAliasRegExData(string expressionName,
  long siteId, string expression, string expressionMap,
  long regexId, string transformedUrl, bool isEnabled,
  EkEnumeration.RegExPriority priority, System.DateTime timeStamp)
```

User

8.50 and higher

The User manager category manages user information and has the following classes:

- *UserGroupManager* on the next page. Manages groups.
- *UserManager* on page 1854. Manages users.

UserGroupManager

8.50 and higher

The UserGroupManager class manages groups. See Managing Users and User Groups in the Ektron Reference for information about managing users and user groups in Ektron.

Namespace

```
Ektron.Cms.Framework.User.UserGroupManager
```

Constructors

- `UserGroupManager()`
- `UserGroupManager(ApiAccessMode)`

Properties

- `UserGroupService`. Returns an instance of the business objects user service.

Methods

- *Add* below
- *AddUser* on page 1832
- *Delete* on page 1834
- *DeleteUser* on page 1835
- *GetItem* on page 1838
- *GetList* on page 1839
- *GetListForUser (list of groups)* on page 1842
- *GetListForUser (all groups)* on page 1845
- `GetUserList(Int64)`. 8.60 and higher Returns list of users that belongs to the user group.
- *IsUserInGroup* on page 1847
- *Update* on page 1849

Add

```
Add(Ektron.Cms.UserGroupData)
```

Adds a user group with details from the supplied UserGroupData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Group Name

Parameters

- `userGroup`. The UserGroupData object to add to the CMS.

Returns

Added UserGroupData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserGroupNameLabel" AssociatedControlID="uxUserGroupName"
CssClass="span-3 last"
            runat="server" Text="*UserGroupName :" />
        <ektronUI:TextField ID="uxUserGroupName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <div class="ektronTopSpace"></div>
        <li class="clearfix">
        <ektronUI:Label ID="uxIsmembershipGroupLabel"
AssociatedControlID="uxIsmembershipGroup" CssClass="span-4 last"
            runat="server" Text="IsMembershipGroup :" />
        <asp:CheckBox ID="uxIsmembershipGroup" runat="server"/>
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />

</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            if (!string.IsNullOrEmpty(uxUserGroupName.Text))
            {
                UserGroupManager UserGroupmanager = new UserGroupManager();


                //Set the UserGroup Details
                UserGroupData UserGroupdata = new UserGroupData()
                {
                    GroupName = uxUserGroupName.Text,
                    IsMemberShipGroup = uxIsmembershipGroup.Checked,
                };
                //Add a New UserGroup with UserGroup Details
                UserGroupmanager.Add(UserGroupdata);

                MessageUtilities.UpdateMessage(uxMessage, "UserGroup Saved with Id " +
UserGroupdata.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage,  "Please Enter UserGroupName
and try again", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

AddUser

```
AddUser(System.Int64,System.Int64)
```

Add a user to a group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Group ID
- * User ID

Parameters

- `groupId`. ID of the group.
- `userId`. ID of the user.

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
        <ektronUI:Label ID="uxUserGroupIdLabel" AssociatedControlID="uxUserGroupId"
CssClass="span-3 last" runat="server" Text="UserGroup Id:" />
        <ektronUI:TextField ID="uxUserGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last" runat="server" Text="User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Add"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserGroupManager UserGroupmanager = new UserGroupManager();
            //Add a User  to a UserGroup
            UserGroupmanager.AddUser(long.Parse(uxUserGroupId.Text), long.Parse
(uxUserId.Text));

            MessageUtilities.UpdateMessage(uxMessage, "User Id : " + uxUserId.Text + "
has been added to a UserGroup Id :  " + uxUserGroupId.Text,
Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
```

```
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Delete

```
Delete(System.Int64)
```

Deletes a UserGroup from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Group ID

Parameters

- `id`. ID of the user group to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserGroupIdLabel" AssociatedControlID="uxUserGroupId"
CssClass="span-3 last" runat="server" Text="*UserGroup Id:" />
        <ektronUI:TextField ID="uxUserGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserGroupManager UserGroupmanager = new UserGroupManager();
            long UserGroupId = long.Parse(uxUserGroupId.Text);
            if (UserGroupId > 0)
            {
                UserGroupData usergroupdata = UserGroupmanager.GetItem(UserGroupId);
                if (usergroupdata != null)
                {
                    //Delete the UserGroup
                    UserGroupmanager.Delete(UserGroupId);
                    MessageUtilities.UpdateMessage(uxMessage, " UserGroup Id : " +
UserGroupId + " has been Deleted from CMS ", Message.DisplayModes.Success);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage,  "Please Enter a Valid
UserGroup ID", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter a Valid
UserGroup ID", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

DeleteUser

```
DeleteUser(System.Int64,System.Int64)
```

Deletes a user from a group in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Group ID
- * User ID

Parameters

- `groupId`. The identifier of the group.
- `userId`. The identifier of the user.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserGroupIdLabel" AssociatedControlID="uxUserGroupId"
CssClass="span-3 last"
            runat="server" Text="*UserGroup Id:" />
        <ektronUI:TextField ID="uxUserGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserGroupManager UserGroupmanager = new UserGroupManager();
            UserManager Usermanager = new UserManager();

            long UserGroupId = long.Parse(uxUserGroupId.Text);
            long UserId = long.Parse(uxUserId.Text);

            if (UserGroupId > 0)
            {
                UserGroupData usergroupdata = UserGroupmanager.GetItem(UserGroupId);
                if (usergroupdata != null)
                {
                    UserData userdata = Usermanager.GetItem(UserId);
                    if (userdata != null)
                    {
                        UserGroupmanager.DeleteUser(UserGroupId, UserId);
                        MessageUtilities.UpdateMessage(uxMessage, " User has been
Deleted from UserGroup. ", Message.DisplayModes.Success);
                    }
                    else
                    {
                        MessageUtilities.UpdateMessage(uxMessage, "Please enter a Valid
User ID.", Message.DisplayModes.Error);
                    }
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "Please Enter a Valid
UserGroup ID", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter a Valid
UserGroup ID", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetItem

```
GetItem(Ektron.Cms.UserGroupData)
```

Retrieves the properties of a specific UserGroupData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Group ID

Parameters

- `id`. ID of the user group to get.

Returns

UserGroup details in a UserGroupData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserGroupIdLabel" AssociatedControlID="uxUserGroupId"
CssClass="span-3 last"
            runat="server" Text="*UserGroup Id:" />
        <ektronUI:TextField ID="uxUserGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserGroupName" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserGroupManager UserGroupmanager = new UserGroupManager();
            long UserGroupId = long.Parse(uxUserGroupId.Text);

            //Get the UserGroup Details
            UserGroupData UserGroupData = UserGroupmanager.GetItem(UserGroupId);
            if (UserGroupData != null)
            {
                //Display  UserGroup Details
                MessageUtilities.UpdateMessage(uxMessage, "UserGroup Details for
UserGroup Id : " + UserGroupId + " are below", Message.DisplayModes.Success);
                uxUserGroupName.Text = "UserGroupName : " + UserGroupData.GroupName;
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
UserGroupID", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetList

```
GetList(Ektron.Cms.UserGroupCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- User Group Property
- * Object Value

Parameters

- `criteria`. UserGroupCriteria used to specify, or filter, the user groups to return.

Returns

A list of user groups.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserGroupPropertyLabel"
AssociatedControlID="uxUserGroupProperty" CssClass="span-4 last"
            runat="server" Text="UserGroupProperty :" />
        <asp:DropDownList ID="uxUserGroupProperty" runat="server">
            <asp:ListItem>Group Id</asp:ListItem>
            <asp:ListItem>Group Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFilterOperatorLabel" CssClass="span-4 last" runat="server"
            AssociatedControlID="uxFilterOperator" Text="FilterOperator : " />
        <ektronUI:Label ID="uxFilterOperator" CssClass="span-6" runat="server"
Text="GreaterThanAndEqualTo/Contains " />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last"
            runat="server" Text="ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxUserlistView" runat="server" ItemPlaceholderID="aspItemPlaceholder">
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
```

```
                    <th>
                        Group Id
                    </th>
                    <th>
                        Group Name
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                    <%# Eval("GroupId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("GroupName")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Common;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            UserGroupManager UserGroupmanager = new UserGroupManager();
            UserGroupCriteria criteria = new UserGroupCriteria();

            if (uxUserGroupProperty.SelectedItem.Text == "Group Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(UserGroupProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else
```

```
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(UserGroupProperty.Name,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            List<UserGroupData> UserGroupList = UserGroupmanager.GetList(criteria);

            uxUserlistView.Visible = true;
            uxUserlistView.DataSource = UserGroupList;
            uxUserlistView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetListForUser (list of groups)

```
GetListForUser(Ektron.Cms.UserGroupCriteria)
```

Retrieves a list of groups to which a user belongs.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- User Group Property
- * Object Value
- User ID

Parameters

- `userId`. ID of the user.
- `orderBy`. Indicates the group order.

Returns

A list of user groups.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserGroupPropertyLabel"
AssociatedControlID="uxUserGroupProperty"
            CssClass="span-4 last" runat="server" Text="UserGroupProperty :" />
        <asp:DropDownList ID="uxUserGroupProperty" runat="server">
            <asp:ListItem>Group Id</asp:ListItem>
            <asp:ListItem>Group Name</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last"
            runat="server" Text="ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxUserlistView" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.
    </EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        User Id
                    </th>
                    <th>
                        IsMemberShipGroup
                    </th>
                    <th>
                        GroupName
                    </th>
                    <th>
                        GroupId
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
```

```
        <tr>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsMemberShipGroup")%>
            </td>
            <td class="devsite-method">
                <%# Eval("GroupName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("GroupId")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            UserGroupManager UserGroupmanager = new UserGroupManager
(Ektron.Cms.Framework.ApiAccessMode.Admin);
            UserGroupCriteria criteria = new UserGroupCriteria();

            if (uxUserGroupProperty.SelectedItem.Text == "Group Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(UserGroupProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(UserGroupProperty.Name,
CriteriaFilterOperator.Contains, Objectvalue);
            }

            long userId;
            userId = Convert.ToInt64(uxUserId.Text);
            List<UserGroupData> UserGroupList = UserGroupmanager.GetListForUser(userId,
```

```
criteria);

            uxUserlistView.Visible = true;
            uxUserlistView.DataSource = UserGroupList;
            uxUserlistView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetListForUser (all groups)

```
GetListForUser(System.Int64,Ektron.Cms.Common.EkEnumeration.GroupOrderBy)
```

Retrieves all of the groups to which a user belongs.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `userId`. ID of the user.
- `criteria`. Used to specify, or filter, the user groups to return.

Returns

Returns a list of user groups.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-4 last"
            runat="server" Text="User Id :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
```

```
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>

<asp:ListView ID="uxUserlistView" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.
    </EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        User Id
                    </th>
                    <th>
                        GroupId
                    </th>
                    <th>
                        Group Name
                    </th>
                    <th>
                        IsMemberShipGroup
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("UserId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("GroupId")%>
            </td>
            <td class="devsite-method">
                <%# Eval("GroupName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("IsMemberShipGroup")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            long userId;
            UserGroupManager UserGroupmanager = new UserGroupManager
(Ektron.Cms.Framework.ApiAccessMode.Admin);
            UserGroupCriteria criteria = new UserGroupCriteria();

            userId = Convert.ToInt64(uxObjectValue.Text);
            List<UserGroupData> UserGroupList = UserGroupmanager.GetListForUser(userId,
EkEnumeration.GroupOrderBy.GroupName);

            uxUserlistView.Visible = true;
            uxUserlistView.DataSource = UserGroupList;
            uxUserlistView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

IsUserInGroup

```
IsUserInGroup(System.Int64,System.Int64)
```

Determines if the user is in the specified group.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * User Group ID

Parameters

- `userId`. ID of the user.
- `groupId`. ID of the user group.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last"
            runat="server" Text="User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserGroupIdLabel" AssociatedControlID="uxUserGroupId"
CssClass="span-3 last"
            runat="server" Text="UserGroup Id:" />
        <ektronUI:TextField ID="uxUserGroupId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Status"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserGroupManager UserGroupmanager = new UserGroupManager();
            //Check if the user is in this group
            bool Check = UserGroupmanager.IsUserInGroup(long.Parse(uxUserId.Text),
long.Parse(uxUserGroupId.Text));
            if (Check)
            {
```

```
                MessageUtilities.UpdateMessage(uxMessage, "User Id : " + uxUserId.Text +
" exists in the UserGroup Id :  " + uxUserGroupId.Text, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "User Id : " + uxUserId.Text +
" does not exists in the UserGroup Id :  " + uxUserGroupId.Text,
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Update

```
Update(Ektron.Cms.UserGroupData)
```

Updates an existing UserGroupData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Group ID
- User Group Name
- Display Name

Parameters

- `userGroup`. The UserGroupData object to update in the CMS.

Returns

Updated UserGroupData object.

Remarks

Validate the user group item with `GetItem()` before you update the properties. Then, call `Update` with your modified UserGroupData object.

NOTE: You cannot change all properties after the initial Add event, such as the UserGroupData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserGroupIdLabel" AssociatedControlID="uxUserGroupId"
CssClass="span-3 last"
            runat="server" Text="UserGroupId :" />
        <ektronUI:TextField ID="uxUserGroupId" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserGroupNameLabel" AssociatedControlID="uxUserGroupName"
CssClass="span-3 last"
            runat="server" Text="UserGroupName :" />
        <ektronUI:TextField ID="uxUserGroupName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserGroupDisplayNameLabel"
AssociatedControlID="uxUserGroupDisplayName" CssClass="span-3 last"
            runat="server" Text="DisplayName :" />
        <ektronUI:TextField ID="uxUserGroupDisplayName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserGroupManager UserGroupmanager = new UserGroupManager();
            //Set the UserGroup Details
            long GroupId = Convert.ToInt64(uxUserGroupId.Text);
            UserGroupData UserGroupdata = UserGroupmanager.GetItem(GroupId);
            if (UserGroupdata != null)
            {
                UserGroupdata.GroupName = uxUserGroupName.Text != string.Empty ?
uxUserGroupName.Text : UserGroupdata.GroupName;
```

```
                UserGroupdata.GroupDisplayName = uxUserGroupDisplayName.Text !=
string.Empty ? uxUserGroupDisplayName.Text : UserGroupdata.GroupDisplayName;

                //Update UserGroup Details
                UserGroupmanager.Update(UserGroupdata);
                MessageUtilities.UpdateMessage(uxMessage, "UserGroup Details has been
Updated for UserGroupId : " + UserGroupdata.Id, Message.DisplayModes.Success);
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid Group ID",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Data Classes

UserGroupCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms
```

Properties

- `UserGroupCriteria()`

  ```
  public UserGroupCriteria()
  ```

- `UserGroupCriteria(Ektron.Cms.Common.UserGroupProperty, EkEnumeration.OrderByDirection)`

  ```
  public UserGroupCriteria(Ektron.Cms.Common.UserGroupProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `UserGroupProperty` are:

  - CommunityGroupId
  - Domain
  - GroupType
  - Id
  - IsMembershipGroup
  - Name

UserGroupData

Namespace

```
Ektron.Cms
```

Properties

- `DisplayUserName`

  ```
  public string DisplayUserName { set; get; }
  ```

- `Domain`. Gets or sets the domain of the UserGroup for the UserGroupData object. Returns the UserGroup's domain.

  ```
  public string Domain { set; get; }
  ```

- `Email`

  ```
  public string Email { set; get; }
  ```

- `FirstName`

  ```
  public string FirstName { set; get; }
  ```

- `GroupDisplayName`. Gets or sets the UserGroup display name for the UserGroupData object. Returns the UserGroup's Display Name.

  ```
  public string GroupDisplayName { set; get; }
  ```

- `GroupDomain`

  ```
  public string GroupDomain { set; get; }
  ```

- `GroupId`

  ```
  public long GroupId { set; get; }
  ```

- `GroupName`. Gets or sets the name of the UserGroup for the UserGroupData object. Returns the UserGroup's name.

  ```
  public string GroupName { set; get; }
  ```

- `GroupPath`. Gets or sets the path of the UserGroup for the UserGroupData object. Returns the UserGroup's path.

  ```
  public string GroupPath { set; get; }
  ```

- `GroupType`. Gets or sets the Group Type, CMSGroup, MembershipGroup, CommunityGroup, or FriendGroup. Returns a group type integer. Use the `UserGroupProperty.GroupType` for conversion and lookup.

  ```
  public int GroupType { set; get; }
  ```

- `Id`. Gets or sets the Id of the UserGroup for the UserGroupData object. Returns the UserGroup's long ID.

  ```
  public override long Id { set; get; }
  ```

- `IsMemberShipGroup`. Gets or sets whether the UserGroup for the UserGroupData object is a CMS UserGroup or a Membership group.
  - True = UserGroup for the UserGroupData object is a Membership group.
  - False = UserGroup for the UserGroupData object is a CMS UserGroup.

  ```
  public bool IsMemberShipGroup { set; get; }
  ```

- `IsMemberShipUser`

```
public bool IsMemberShipUser { set; get; }
```

- `LastName`

```
public string LastName { set; get; }
```

- `Name`. Gets or sets the name of the UserGroup for the UserGroupData object. Returns the UserGroup's name.

```
public string Name { set; get; }
```

- `UserCount`. Gets or sets the number of users in the UserGroup for the UserGroupData object.

```
public int UserCount { set; get; }
```

- `UserGroupDescription`. Gets or sets the user group description

```
public string UserGroupDescription { set; get; }
```

- `UserId`

```
public long UserId { set; get; }
```

- `UserName`

```
public string UserName { set; get; }
```

UserManager

8.50 and higher

The UserManager class manages users. See Managing Users and User Groups in the Ektron Reference for information about managing users and user groups in Ektron.

Namespace

```
Ektron.Cms.Framework.User.UserManager
```

Constructors

- `UserManager()`
- `UserManager(ApiAccessMode)`

Properties

- `UserService`. Returns an instance of the business objects user service.

Methods

- *ActivateUserAccount (account ID)* on the facing page
- *ActivateUserAccount (user)* on page 1857
- *ActivateUserAccounts (user ID)* on page 1859
- *Add* on page 1861
- *AssignTaxonomy* on page 1864
- *Authenticate* on page 1866
- *UserManager* above
- *CanViewUserProfile* on page 1868
- *Delete* on page 1871
- `GetCustomPropertyList()`. Gets the list of all user custom properties.
- *GetItem (authentication)* on page 1873
- *GetItem (ID)* on page 1875
- `GetItem(Int64,Boolean,Boolean)`. 9.00 and higher Returns details of a user.
- *GetItem (properties)* on page 1877
- *GetList (property)* on page 1879
- *GetList (taxonomy)* on page 1882
- *GetList (user)* on page 1885
- `LockUser(Int64)`. 8.60 and higher Locks the user account by user ID.
- *Login (automatic)* on page 1888

- *Login (user)* on page 1890
- `Logout()`. Log out the current user.

  NOTE: The user will need to refresh the page.

- *RemoveTaxonomy* on page 1892
- `ResetPassword(String,Int64)`. 8.60 and higher Resets the membership user password by user name and message ID. You can specify which workarea-managed message you want to use. For example, you could have a different reset password message for your intranet versus internet sites. The messages are managed in **Workarea > Settings > Community Management > Messages**.
- `ResetPassword(String,String,String)`. 8.60 and higher Resets the user password by user name, old password and new password.
- `UnlockUser(Int64)`. 8.60 and higher Unlocks the user account by user ID.
- *Update* on page 1893

ActivateUserAccount (account ID)

8.60 and higher

```
ActivateUserAccount(System.String,System.String)
```

Activates a deactivated user into the current or active state using the username and account ID.

Background Information

Ektron supports 2 types of users:

- CMS users (licensed authors)
- Membership users (site members)

You can configure Ektron so that self-registering membership users must be validated by a site manager/moderator before they can use their account. ActivateUserAccount verifies and activates a membership account.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Name
- * Account ID

Parameters

- `userName`. The user name login for the user.
- `accountId`. The account ID for the user.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronui:label id="uxUserNameLabel" associatedcontrolid="uxUserName"
cssclass="span-3 last"
            runat="server" text="*User Name:" />
        <ektronui:textfield id="uxUserName" cssclass="span-6" runat="server"
validationgroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronui:label id="uxAccountIdLabel" associatedcontrolid="uxAccountId"
cssclass="span-3 last"
            runat="server" text="*Account Id:" />
        <ektronui:textfield id="uxAccountId" cssclass="span-6" runat="server"
validationgroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronui:label id="uxStatus" visible="false" cssclass="span-9 last"
runat="server"
            text="Status:" />
    </li>
    <li class="clearfix">
        <ektronui:button id="uxSubmit" runat="server" onclick="uxSubmit_Click"
text="Activate"></ektronui:button>
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using Ektron.Cms;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.User;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager userManager = new UserManager();
            long userId = long.Parse(uxUserId.Text);

            userManager.ActivateUserAccount(userId);

            UserData userData = userManager.GetItem(userId);
            if (userData != null)
```

```
            {
                if (!userData.IsDeleted)
                {
                    MessageUtilities.UpdateMessage(uxMessage, " The User Id " + userId +
" has been activated. ", Message.DisplayModes.Success);
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, " The User Id is not
activated. ", Message.DisplayModes.Error);
                    return;
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please enter a Valid User
Id.", Message.DisplayModes.Error);
                return;
            }

        uxPageMultiView.SetActiveView(uxViewMessage);
    }
    catch (Exception ex)
    {
        MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
        uxPageMultiView.SetActiveView(uxViewMessage);
    }

    }
```

ActivateUserAccount (user)

8.60 and higher

```
ActivateUserAccount(System.String,System.String)
```

Activates a deactivated user into the current or active state using the user ID.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Name

Parameters

- `userId`. A unique ID to activate.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronui:label id="uxUserIdLabel" associatedcontrolid="uxUserId"
cssclass="span-3 last"
            runat="server" text="*User Id:" />
        <ektronui:textfield id="uxUserId" cssclass="span-6" runat="server"
validationgroup="RegisterValidationGroup"
            text="0" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronui:label id="uxStatus" visible="false" cssclass="span-9 last"
runat="server"
            text="Status:" />
    </li>
    <li class="clearfix">
        <ektronui:button id="uxSubmit" runat="server" onclick="uxSubmit_Click"
text="Activate"></ektronui:button>
    </li>
</ol>
<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using Ektron.Cms;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.User;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager userManager = new UserManager();
            String userName = uxUserName.Text;
            String accountId = uxAccountId.Text;

            userManager.ActivateUserAccount(userName, accountId);

            Ektron.Cms.User.UserCriteria userCriteria = new Ektron.Cms.User.UserCriteria
();
            userCriteria.AddFilter(Ektron.Cms.User.UserProperty.UserName,
Ektron.Cms.Common.CriteriaFilterOperator.EqualTo, userName);

            List<UserData> userDataList = userManager.GetList(userCriteria);

            foreach (UserData userData in userDataList)
            {

                if (userData != null)
                {
                    if (!userData.IsDeleted)
```

```
                    {
                        MessageUtilities.UpdateMessage(uxMessage, " The User Name " +
userName + " has been activated. ", Message.DisplayModes.Success);
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, " The User Name is not
activated ", Message.DisplayModes.Error);
                        return;
                    }
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please enter a Valid User
Name and Account Id.", Message.DisplayModes.Error);
                    return;
                }
            }

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

ActivateUserAccounts (user ID)

8.60 and higher

```
ActivateUserAccounts(System.Collections.Generic.List)
```

Activates the deactivated users into the current or active state using the user IDs.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Name
- * Account ID

Parameters

- `userIds`. The list of user IDs to activate.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronui:label id="uxUserIdsLabel" associatedcontrolid="uxUserIds"
cssclass="span-5 last"
            runat="server" text="*User Ids (Comma Seperated):" />
        <ektronui:textfield id="uxUserIds" cssclass="span-6" runat="server"
validationgroup="RegisterValidationGroup"
            text="0" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronui:label id="uxStatus" visible="false" cssclass="span-9 last"
runat="server"
            text="Status:" />
    </li>
    <li class="clearfix">
        <ektronui:button id="uxSubmit" runat="server" onclick="uxSubmit_Click"
text="Activate"></ektronui:button>
    </li>
</ol>
<asp:literal id="uxMessage" runat="server"></asp:literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.User;
using Ektron.Site.Developer;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager userManager = new UserManager();
            string ids = uxUserIds.Text;
            string[] userIds = ids.Split(new char[] { ',' },
StringSplitOptions.RemoveEmptyEntries);
            if (userIds.Length > 0)
            {
                List<long> userIdList = new List<long>();
                foreach (string userId in userIds)
                {
                    userIdList.Add(long.Parse(userId));
                }
                userManager.ActivateUserAccounts(userIdList);
                MessageUtilities.UpdateMessage(uxMessage, " The User Ids " + ids + " has
been activated. ", Message.DisplayModes.Success);
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please enter a User Id.",
```

```
Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Add

```
Add(Ektron.Cms.UserData)
```

Adds a user with details from the supplied UserData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Name
- * Password
- * First Name
- * Last Name
- * Display
- Email

Parameters

- `userData`. The UserData object to add to the CMS.

Returns

Returns the custom CmsData object added.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserNameLabel" AssociatedControlID="uxUserName"
CssClass="span-3 last"
            runat="server" Text="*UserName :" />
        <ektronUI:TextField ID="uxUserName" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
```

```
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordLabel" AssociatedControlID="uxPassword"
CssClass="span-3 last"
            runat="server" Text="*Password :" />
        <asp:TextBox ID="uxPassword" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            TextMode="Password" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFirstNameLabel" AssociatedControlID="uxFirstName"
CssClass="span-3 last"
            runat="server" Text="*FirstName :" />
        <ektronUI:TextField ID="uxFirstName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLastNameLabel" AssociatedControlID="uxLastName"
CssClass="span-3 last"
            runat="server" Text="*LastName :" />
        <ektronUI:TextField ID="uxLastName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDisplayLabel" AssociatedControlID="uxDisplay"
CssClass="span-3 last"
            runat="server" Text="*Display :" />
        <ektronUI:TextField ID="uxDisplay" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEmailLabel" AssociatedControlID="uxEmail" CssClass="span-3
last"
            runat="server" Text="Email :" />
        <ektronUI:TextField ID="uxEmail" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Save"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
```

```
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager userManager = new UserManager();
            //Check the Valid EmailAddress or Not
            string emailAddress = uxEmail.Text;
            if (!string.IsNullOrEmpty(emailAddress))
            {
                try
                {
                    System.Net.Mail.MailAddress addr = new System.Net.Mail.MailAddress
(emailAddress);
                }
                catch
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Email
Address.", Message.DisplayModes.Error);
                    return;
                }
            }

            //long TIME_ZONE = 240;
            //long PHONE = 242;

            //Set the User Details
            UserData userData = new UserData()
            {
                Username = uxUserName.Text,
                Password = uxPassword.Text,
                FirstName = uxFirstName.Text,
                LastName = uxLastName.Text,
                DisplayName = uxDisplay.Text,
                Email = emailAddress,
                IsMemberShip = true
            };

            userData.CustomProperties = userManager.GetCustomPropertyList();
            userData.CustomProperties["Phone"].Value = "603-555-9999";
            userData.CustomProperties["Time Zone"].Value = "Eastern Standard Time";
            //userData.AttributeList = new CustomAttribute[2];
            //userData.AttributeList[0] = new CustomAttribute() { ID = "242", Value =
"603-555-1212" };
            //userData.AttributeList[1] = new CustomAttribute() { ID = "240", Value =
"Hawaiian Standard Time" };

            //Add a New User with User Details
            userManager.Add(userData);
            MessageUtilities.UpdateMessage(uxMessage, "User Saved with Id " +
userData.Id, Message.DisplayModes.Success);
```

```
            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

AssignTaxonomy

```
AssignTaxonomy(System.Int64,System.Int64)
```

Assigns a user taxonomy to a category.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Taxonomy ID

Parameters

- `userId`. ID of the user.
- `categoryId`. ID of the category.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last" runat="server" Text="User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />

    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxTaxonomyIDLabel" AssociatedControlID="uxTaxonomyID"
CssClass="span-3 last" runat="server" Text="Taxonomy Id:" />
        <ektronUI:TextField ID="uxTaxonomyID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />

    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
```

```
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Save"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxUserId.Text);
            long TaxonomyID = long.Parse(uxTaxonomyID.Text);

            //Get the User data to check Valid UserId or Not
            UserData Userdata = Usermanager.GetItem(UserId);
            if (Userdata != null)
            {
                TaxonomyManager taxonomyManager = new TaxonomyManager();
                TaxonomyData taxonomyData = taxonomyManager.GetItem(TaxonomyID);

                if (taxonomyData != null)
                {
                    //Assign User Taxonomy to a Category
                    Usermanager.AssignTaxonomy(UserId, TaxonomyID);
                    MessageUtilities.UpdateMessage(uxMessage,  " User Id : " + UserId +
" has been Assigned to Taxonomy ID : " + TaxonomyID, Message.DisplayModes.Success);
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please enter a Valid
Taxonomy ID.", Message.DisplayModes.Error);
                    return;
                }
            }
            else
            {
```

```
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please enter a Valid User
ID.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Authenticate

Authenticate a user:

```
Authenticate(System.String,System.String)
```

Authenticate a user in a specific domain:

```
Authenticate(System.String,System.String,System.String)
```

Retrieves an authentication token that you can use to authenticate the user in a particular domain. WCF headers pass the authentication token to the application tier.

NOTE: Authenticate is used only in 3-Tier deployments.

In 2-Tier deplolyments, if you need to run an API using elevated permissions, you can initialize the API to put it into "Internal Admin" mode.

```
var UserCRUD = new UserManager(Ektron.Cms.Framework.ApiAccessMode.Admin);
```

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Name
- * Password
- Domain (Optional for user in a specific domain)

Parameters

- `userName`. The user name of user to validate.
- `password`. The user's password.
- `domain`. Optional: The domain to validate.

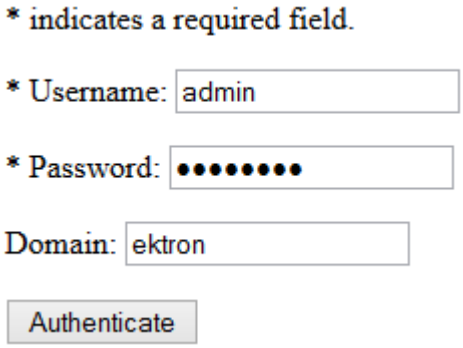
Returns

An authentication token. If authentication fails, an empty string is returned.

NOTE: Authenticate does not log the user into the CMS. Use the *Login (user)* on page 1890 method to log in a user.

.aspx code snippet

```
<asp:Panel ID="uxAuthenticateForm" runat="server" DefaultButton="uxAuthenticate">
    <p>
        * indicates a required field.
    </p>
    <p>
        * Username:
        <asp:TextBox ID="uxUsername" runat="server" />
    </p>
    <p>
        * Password:
        <asp:TextBox ID="uxPassword" runat="server" TextMode="Password" />
    </p>
    <p>
        Domain:
        <asp:TextBox ID="uxDomain" runat="server" />
    </p>
    <p>
        <asp:Button ID="uxAuthenticate" runat="server" Text="Authenticate"
OnClick="uxAuthenticate_Click" />
    </p>
</asp:Panel>
<asp:Panel ID="uxAuthenticateFail" runat="server" Visible="false">
    Authentication failed for the supplied username, password, and domain.
</asp:Panel>
<asp:Panel ID="uxAuthenticateSuccess" runat="server" Visible="false">
    Your Authentication Token is: <asp:Literal ID="uxToken" runat="server" />
</asp:Panel>
```



Pass: `Your Authentication Token is: 5c80b4dc7a9947949c749f7ae54319d1`

Fail: `Authentication failed for the supplied username, password, and domain.`

.aspx.cs code-behind method

```
protected void uxAuthenticate_Click(object sender, EventArgs e)
{
    uxAuthenticateForm.Visible = false;
    var UserCRUD = new Ektron.Cms.Framework.User.UserManager();
    string Token = UserCRUD.Authenticate(uxUsername.Text, uxPassword.Text,
uxDomain.Text);
    if (!string.IsNullOrEmpty(Token)) // Success
    {
        uxAuthenticateSuccess.Visible = true;
        uxToken.Text = Token;
    }
    else // Fail
    {
        uxAuthenticateFail.Visible = true;
    }
}
```

Sample Token Use

```
private void UpdateContent(Ektron.Cms.ContentData ContentToUpdate, string
AuthenticationToken)
{
    var ContentCRUD = new Ektron.Cms.Framework.Content.ContentManager();
    if (!string.IsNullOrEmpty(AuthenticationToken))
    {
        string originalToken = ContentCRUD.RequestInformation.AuthenticationToken;
        try
        {
            ContentCRUD.RequestInformation.AuthenticationToken = AuthenticationToken;
            ContentCRUD.Update(ContentToUpdate);
        }
        catch (Exception ex)
        {
            Ektron.Cms.EkException.LogException(ex);
        }
        finally
        {
            ContentCRUD.RequestInformation.AuthenticationToken = originalToken;
        }
    }
}
```

CanViewUserProfile

```
CanViewUserProfile(System.Int64,System.Int64)
```

Returns true if the viewer (viewerId) can view the other users profile (userid).

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Viewer ID

Parameters

- `userId`. ID of user's profile to view.
- `viewerId`. ID of user trying to view profile.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxViewerIdLabel" AssociatedControlID="uxViewerId"
CssClass="span-3 last"
            runat="server" Text="*Viewer Id:" />
        <ektronUI:TextField ID="uxViewerId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
        <div class="ektronTopSpace"></div>
    <li class="clearfix">
        <ektronUI:Message ID="uxStatus" DisplayMode="Error" runat="server"
Visible="false" />
    </li>
        <div class="ektronTopSpace"></div>
        <li class="clearfix">
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="CanView"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" /></li>
</ol>

<ektronUI:Message ID="uxMessage" DisplayMode="Information" Visible="false"
runat="server" />
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
```

```
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Framework.Organization;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxUserId.Text);
            long ViewerId = long.Parse(uxViewerId.Text);
            bool status = false;
            //Get the User data to check Valid UserId or Not
            UserData Userdata = Usermanager.GetItem(UserId);
            if (Userdata != null)
            {
                //Get the Viewer data to check Valid ViewerId or Not
                UserData viewerdata = Usermanager.GetItem(ViewerId);

                if (viewerdata != null)
                {
                    status = Usermanager.CanViewUsersProfile(UserId, ViewerId);
                    if (status)
                        MessageUtilities.UpdateMessage(uxMessage, " Viewer Id : " +
ViewerId + " can view the user Id : " + UserId + " profile.",
Message.DisplayModes.Success);
                    else
                        MessageUtilities.UpdateMessage(uxMessage, " Viewer Id : " +
ViewerId + " Cannot view the user Id : " + UserId + " profile.",
Message.DisplayModes.Success);
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please enter a Valid
Viewer ID.", Message.DisplayModes.Error);
                    return;
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please enter a Valid User
ID.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
```

```
    }
```

Delete

```
Delete(System.Int64)
```

Deletes a user (using the UserData object) from the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `id`. ID of the user to delete.

Remarks

Validate the item with `GetItem()` before deleting it.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last" runat="server" Text="User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
        <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Delete"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxUserId.Text);
            if (UserId > 0)
            {
                UserData userdata = Usermanager.GetItem(UserId);
                if (userdata != null)
                {
                    if (!userdata.IsDeleted)
                    {   //Delete the UserID
                        Usermanager.Delete(UserId);
                        MessageUtilities.UpdateMessage(uxMessage,  " User Id : " +
UserId + " has been Deleted from CMS ", Message.DisplayModes.Success);
                    }
                    else
                    {
                        uxStatus.Visible = true;
                        MessageUtilities.UpdateMessage(uxStatus, "Entered User ID is
already deleted.", Message.DisplayModes.Error);
                        return;
                    }
                }
                else
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please enter a Valid User
ID.", Message.DisplayModes.Error);
                    return;
                }
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please enter a Valid User
ID.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetItem (authentication)

```
GetItem(System.Int64,System.String)
```

Retrieves details of a user, based on an authentication type and authentication user ID.

Authentication Types represent third-party integrations for authenticating users. For example, `AuthenticationTypeId = 1` is Facebook integration and the `authenticationUserId` in this case would be the users Facebook ID. So `GetItem (1, '1111111111')` would return the CMS user who logs in through Facebook and has the Facebook ID '1111111111'.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Authentication Type ID
- Authentication User ID

Parameters

- `authenticationTypeId`. ID of authentication type user is associated with.
- `authenticationUserId`. ID of user based on authentication type.

Returns

User details in a UserData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxAuthenticationTypeIDLabel"
AssociatedControlID="uxAuthenticationTypeID" CssClass="span-4 last"
            runat="server" Text="AuthenticationType ID:" />
        <ektronUI:TextField ID="uxAuthenticationTypeID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
        <li class="clearfix">
        <ektronUI:Label ID="uxAuthenticationUserIDLabel"
AssociatedControlID="uxAuthenticationUserID" CssClass="span-4 last"
            runat="server" Text="AuthenticationUser ID:" />
        <ektronUI:TextField ID="uxAuthenticationUserID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix" style="color: red;">
```

```
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetDetails"></ektronUI:Button>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFirstName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLastName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDisplayName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxEmail" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager Usermanager = new UserManager();
            long AuthenticationTypeID = long.Parse(uxAuthenticationTypeID.Text);
            //Get the User data for given ID
            UserData Userdata = Usermanager.GetItem(AuthenticationTypeID,
uxAuthenticationUserID.Text);
            if (Userdata != null)
            {
                //Display  User Details
                MessageUtilities.UpdateMessage(uxMessage, "User Details for  User Id : "
```

```
+ Userdata.Id + " are below", Message.DisplayModes.Success);

                uxUserName.Text = "UserName : " + Userdata.Username;
                uxFirstName.Text = "FirstName : " + Userdata.FirstName;
                uxLastName.Text = "LastName : " + Userdata.LastName;
                uxDisplayName.Text = "DisplayName : " + Userdata.DisplayName;
                uxEmail.Text = "Email : " + Userdata.Email;
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid
Authentication Values.", Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetItem (ID)

```
GetItem(System.Int64)
```

Retrieves the properties of a specific UserData object.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID

Parameters

- `id`

Returns

User details in a UserData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
```

```
CssClass="span-3 last"
            runat="server" Text="User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetDetails"></ektronUI:Button>
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFirstName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLastName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDisplayName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxEmail" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxUserId.Text);
            //Get the User data for given UserID
            UserData Userdata = Usermanager.GetItem(UserId);
            if (Userdata != null)
            {
                //Display  User Details
```

```
                MessageUtilities.UpdateMessage(uxMessage, "User Details for  User Id : "
+ UserId + " are below", Message.DisplayModes.Success);
                uxUserName.Text = "UserName : " + Userdata.Username;
                uxFirstName.Text = "FirstName : " + Userdata.FirstName;
                uxLastName.Text = "LastName : " + Userdata.LastName;
                uxDisplayName.Text = "DisplayName : " + Userdata.DisplayName;
                uxEmail.Text = "Email : " + Userdata.Email;
                uxEmail.Text = "IsDeleted : " + Userdata.IsDeleted;
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid User ID
and Try again.", Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetItem (properties)

```
GetItem(System.Int64,System.Boolean)
```

Retrieves the properties of a specific UserData object, including any custom properties.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Include Custom Properties

Parameters

- `id`. ID of the user to get.

Returns

User details in a UserData object.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
```

```
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last"
            runat="server" Text="*User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxIncludeCustomLabel" AssociatedControlID="uxIncludeCustom"
CssClass="span-3 last"
            runat="server" Text="*Include Custom Properties:" />
        <ektronUI:Button ID="uxIncludeCustom" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="" DisplayMode="Checkbox" Checked="True" />
    </li>
    <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetItem"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<ol class="formFields">
    <li class="clearfix">
        <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxUserName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxFirstName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxLastName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxDisplayName" runat="server"></asp:Literal>
    </li>
    <li class="clearfix">
        <asp:Literal ID="uxEmail" runat="server"></asp:Literal>
    </li>
        <li class="clearfix">
        <asp:Literal ID="uxTime" runat="server"></asp:Literal>
    </li>
        <li class="clearfix">
        <asp:Literal ID="uxPhone" runat="server"></asp:Literal>
    </li>
</ol>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxUserId.Text);
            bool include = false;
            if (uxIncludeCustom.Checked)
            {
                include = true;
            }

            //Get the User data for given UserID
            UserData Userdata = Usermanager.GetItem(UserId, include);
            if (Userdata != null)
            {
                //Display  User Details
                MessageUtilities.UpdateMessage(uxMessage, "User Details for  User Id : "
+ UserId + " are below", Message.DisplayModes.Success);

                uxUserListView.DataSource = new UserData[]{Userdata};
                uxUserListView.DataBind();
            }
            else
            {
                uxStatus.Visible = true;
                MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid User ID.",
Message.DisplayModes.Error);
                return;
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetList (property)

8.60 and higher

```
GetList(Ektron.Cms.User.UserCustomPropertyCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User Custom Property Name
- * User Custom Property Value

Parameters

- `criteria`. UserCustomPropertyCriteria used to specify, or filter, the users to return.

Returns

A list of users.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserCustomPropertyLabel"
AssociatedControlID="uxUserCustomProperty"
            CssClass="span-6 last" runat="server" Text="*UserCustomProperty Name :" />
        <ektronUI:TextField ID="uxUserCustomProperty" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="Phone" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-6 last"
            runat="server" Text="*UserCustomProperty Value :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="603" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

```
<asp:ListView ID="uxUserlistView" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Name
                    </th>
                    <th>
                        Display Name
                    </th>
                    <th>
                        Email
                    </th>
                </tr>
            </thead>
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("DisplayName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Email")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
```

```
using Ektron.Cms.Framework.User;
using Ektron.Cms.Common;
using Ektron.Cms.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {

            UserManager Usermanager = new UserManager();
            Ektron.Cms.User.UserCustomPropertyCriteria userCustomproperty = new
UserCustomPropertyCriteria();
            object objectvalue= uxObjectValue.Text;
            userCustomproperty.AddFilter(uxUserCustomProperty.Text,
CriteriaFilterOperator.Contains, objectvalue);

            List<UserData> UserList = Usermanager.GetList(userCustomproperty);

            uxUserlistView.Visible = true;
            uxUserlistView.DataSource = UserList;
            uxUserlistView.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetList (taxonomy)

```
GetList(Ektron.Cms.User.UserTaxonomyCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- User Property
- * Object Value

Parameters

- `criteria`. UserTaxonomyCriteria used to specify, or filter, the users to return.

Returns

A list of users.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserPropertyLabel" AssociatedControlID="uxUserProperty"
CssClass="span-3 last"
            runat="server" Text="UserProperty :" />
        <asp:DropDownList ID="uxUserProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>Path</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-3 last"
            runat="server" Text="*ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <li class="clearfix">
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
        <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
    </li>
</ol>

<asp:View ID="uxViewMessage" runat="server">
    <asp:Literal ID="uxMessage" runat="server"></asp:Literal>
</asp:View>
    <asp:ListView ID="uxUserlist" runat="server" ItemPlaceholderID="aspItemPlaceholder"
        Visible="false">
        <EmptyDataTemplate>
            Criteria did not match any records.</EmptyDataTemplate>
        <LayoutTemplate>
            <table class="devsite-api-method">
                <thead>
                    <tr>
                        <th>
                            Id
                        </th>
                        <th>
                            User Name
                        </th>
                        <th>
```

```
                            Display Name
                        </th>
                        <th>
                            Email
                        </th>
                    </tr>
                </thead>
                <tbody>
                    <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
                </tbody>
            </table>
        </LayoutTemplate>
        <ItemTemplate>
            <tr>
                <td class="devsite-method">
                    <%# Eval("Id")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("UserName")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("DisplayName")%>
                </td>
                <td class="devsite-method">
                    <%# Eval("Email")%>
                </td>
            </tr>
        </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Common;
using Ektron.Cms.User;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            string Property = uxUserProperty.SelectedItem.Text;
```

```
            UserManager Usermanager = new UserManager();
            UserTaxonomyCriteria criteria = new UserTaxonomyCriteria(UserProperty.Id,
EkEnumeration.OrderByDirection.Ascending);
            if (Property == "Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(UserTaxonomyProperty.Id,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(UserTaxonomyProperty.Path,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            List<UserData> UserList = Usermanager.GetList(criteria);

            uxUserlist.Visible = true;
            uxUserlist.DataSource = UserList;
            uxUserlist.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

GetList (user)

```
GetList(Ektron.Cms.User.UserCriteria)
```

Retrieves lists of objects.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- User Property
- * Object Value

Parameters

- `criteria`. UserCriteria used to specify, or filter, the users to return.

Returns

A list of users.

See *Criteria use for GetList methods* on page 156 for additional information.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserPropertyLabel" AssociatedControlID="uxUserProperty"
CssClass="span-3 last"
            runat="server" Text="UserProperty :" />
        <asp:DropDownList ID="uxUserProperty" runat="server">
            <asp:ListItem>Id</asp:ListItem>
            <asp:ListItem>UserName</asp:ListItem>
            <asp:ListItem>DisplayName</asp:ListItem>
        </asp:DropDownList>
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxObjectValueLabel" AssociatedControlID="uxObjectValue"
CssClass="span-3 last"
            runat="server" Text="*ObjectValue :" />
        <ektronUI:TextField ID="uxObjectValue" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="1" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="GetList"></ektronUI:Button>
    <ektronUI:Label ID="uxRequiredLabel" CssClass="span-4" runat="server" Text="* -
Required" />
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
<asp:ListView ID="uxUserlist" runat="server" ItemPlaceholderID="aspItemPlaceholder"
    Visible="false">
    <EmptyDataTemplate>
        Criteria did not match any records.</EmptyDataTemplate>
    <LayoutTemplate>
        <table class="devsite-api-method">
            <thead>
                <tr>
                    <th>
                        Id
                    </th>
                    <th>
                        User Name
                    </th>
                    <th>
                        Display Name
                    </th>
                    <th>
                        Email
                    </th>
                </tr>
            </thead>
```

```
            <tbody>
                <asp:PlaceHolder ID="aspItemPlaceholder"
runat="server"></asp:PlaceHolder>
            </tbody>
        </table>
    </LayoutTemplate>
    <ItemTemplate>
        <tr>
            <td class="devsite-method">
                <%# Eval("Id")%>
            </td>
            <td class="devsite-method">
                <%# Eval("UserName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("DisplayName")%>
            </td>
            <td class="devsite-method">
                <%# Eval("Email")%>
            </td>
        </tr>
    </ItemTemplate>
</asp:ListView>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
using Ektron.Cms.Common;
using Ektron.Cms.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            object Objectvalue;
            string Property = uxUserProperty.SelectedItem.Text;

            UserManager Usermanager = new UserManager();
            UserCriteria criteria = new UserCriteria(UserProperty.UserName,
EkEnumeration.OrderByDirection.Ascending);
            if (Property == "Id")
            {
                Objectvalue = Convert.ToInt64(uxObjectValue.Text);
                criteria.AddFilter(UserProperty.Id, CriteriaFilterOperator.EqualTo,
Objectvalue);
            }
```

```
            else if (Property == "UserName")
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(UserProperty.UserName,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }
            else
            {
                Objectvalue = uxObjectValue.Text;
                criteria.AddFilter(UserProperty.DisplayName,
CriteriaFilterOperator.EqualTo, Objectvalue);
            }

            List<UserData> UserList = Usermanager.GetList(criteria);

            uxUserlist.Visible = true;
            uxUserlist.DataSource = UserList;
            uxUserlist.DataBind();

            uxPageMultiView.SetActiveView(uxViewMessage);

        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Login (automatic)

```
Login(System.Int64,System.String)
```

Automatically logs in a user based on an authentication type and authentication user ID.

NOTE: Prior to Ektron 9.40 SP1, this method required you to run `setAuthenticationCookie` (with name `ecm...`) after calling it. In Ektron 9.40 SP1, this is done for you.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Authentication Type ID
- * Authentication User ID

Parameters

- `authenticationTypeId`. ID of authentication type user is associated with.
- `authenticationUserId`. ID of user based on authentication type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxStatusLabel" CssClass="span-6" runat="server" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAuthenticationTypeIDLabel"
AssociatedControlID="uxAuthenticationTypeID"
            CssClass="span-4 last" runat="server" Text="AuthenticationType ID:" />
        <ektronUI:TextField ID="uxAuthenticationTypeID" CssClass="span-6" runat="server"
            ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxAuthenticationUserIDLabel"
AssociatedControlID="uxAuthenticationUserID"
            CssClass="span-4 last" runat="server" Text="AuthenticationUser ID:" />
        <ektronUI:TextField ID="uxAuthenticationUserID" CssClass="span-6" runat="server"
            ValidationGroup="RegisterValidationGroup" Text="0" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Login"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager Usermanager = new UserManager();
            if (!Ektron.Cms.Framework.Context.UserContextService.Current.IsLoggedIn)
            {
                UserData userdata = Usermanager.Login(uxUserName.Text, uxPassword.Text);

                if (userdata != null)
                {
```

```
                    MessageUtilities.UpdateMessage(uxMessage,  " User : " +
uxUserName.Text + " has successfully Logged In ", Message.DisplayModes.Success);
                    Response.Redirect(Request.UrlReferrer.AbsoluteUri);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, " User : " +
Usermanager.RequestInformation.LoggedInUsername + " is already  Logged In ",
Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            //error logging in
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Login (user)

```
Login(System.String,System.String,System.String)
```

Logs in a user.

NOTE: Prior to Ektron 9.40 SP1, this method required you to run `setAuthenticationCookie` (with name `ecm`..) after calling it. In Ektron 9.40 SP1, this is done for you.

NOTE: In Ektron 8.6, this method was modified to generate an error if a user cannot successfully log in. (Previous to 8.6, Login returned an empty UserData object on unsuccessful login attempts, which left the developer unaware of what happened or why it happened.)

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * Username
- * Password

Parameters

- `userName`. User name of user to login.
- `password`. User's password.

- `domain`. Name of domain the user is logging in to. This parameter is only applicable for AD and LDAP configurations.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxStatusLabel" CssClass="span-6" runat="server" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserNameLabel" AssociatedControlID="uxUserName"
CssClass="span-3 last"
            runat="server" Text="UserName :" />
        <ektronUI:TextField ID="uxUserName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"  />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordLabel" AssociatedControlID="uxPassword"
CssClass="span-3 last"
            runat="server" Text="Password :" />
        <asp:TextBox ID="uxPassword" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            TextMode="Password"  />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Login"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager Usermanager = new UserManager();
            if (!Ektron.Cms.Framework.Context.UserContextService.Current.IsLoggedIn)
            {
                //Logs in, a user
                UserData userdata = Usermanager.Login(Convert.ToInt64
(uxAuthenticationTypeID.Text), uxAuthenticationUserID.Text);

                if (userdata.Id > 0)
```

```
                {
                    MessageUtilities.UpdateMessage(uxMessage, " User : " +
userdata.Username + " has successfully Logged In ", Message.DisplayModes.Success);
                    Response.Redirect(Request.UrlReferrer.AbsoluteUri);
                }
                else
                {
                    MessageUtilities.UpdateMessage(uxMessage, "The UserName or Password
you entered is not correct", Message.DisplayModes.Error);
                }
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, " User : " +
Usermanager.RequestInformation.LoggedInUsername + " is already  Logged In ",
Message.DisplayModes.Success);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

RemoveTaxonomy

```
RemoveTaxonomy(System.Int64,System.Int64)
```

Removes a user taxonomy from a category.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- * Taxonomy ID

Parameters

- `userId`. ID of the user.
- `categoryId`. ID of the category.

.aspx code snippet

```
<ol class="formFields">
        <li class="clearfix">
```

```
            <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last" runat="server" Text="User Id:" />
            <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
        </li>
            <li class="clearfix">
            <ektronUI:Label ID="uxTaxonomyIDLabel" AssociatedControlID="uxTaxonomyID"
CssClass="span-3 last" runat="server" Text="Taxonomy Id:" />
            <ektronUI:TextField ID="uxTaxonomyID" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" Text="0" />
        </li>
        <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Save"></ektronUI:Button>
    </ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager Usermanager = new UserManager();
            long UserId = long.Parse(uxUserId.Text);
            long TaxonomyID = long.Parse(uxTaxonomyID.Text);
            //Remove User Taxonomy to a Category
            Usermanager.RemoveTaxonomy(UserId, TaxonomyID);

            MessageUtilities.UpdateMessage(uxMessage, " User Id : " + UserId + " has
been Removed from Taxonomy ID : " + TaxonomyID, Message.DisplayModes.Success);

            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Update

```
Update(Ektron.Cms.UserData)
```

Updates an existing UserData object in the CMS.

Authenticated users

- CMS Administrators

To create anonymous user access with this method, set the API access mode (`ApiMode`) in the Manager class when instantiating the class.

Fields

*=Required

- * User ID
- User Name
- * Password
- First Name
- Last Name
- Display
- Email

Parameters

- `userData`. The UserData object to update in the CMS.

Returns

Returns the custom CmsData object updated.

Remarks

Validate the user item with `GetItem()` before you update the properties. Then, call `Update` with your modified UserData object.

NOTE: You cannot change all properties after the initial Add event, such as the UserData.Type.

.aspx code snippet

```
<ol class="formFields">
    <li class="clearfix">
        <ektronUI:Label ID="uxUserIdLabel" AssociatedControlID="uxUserId"
CssClass="span-3 last"
            runat="server" Text="User Id:" />
        <ektronUI:TextField ID="uxUserId" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup"
            Text="0" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxUserNameLabel" AssociatedControlID="uxUserName"
CssClass="span-3 last"
            runat="server" Text="UserName :" />
        <ektronUI:TextField ID="uxUserName" CssClass="span-4" runat="server"
```

```
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxPasswordLabel" AssociatedControlID="uxPassword"
CssClass="span-3 last"
            runat="server" Text="*Password :" />
        <asp:TextBox ID="uxPassword" CssClass="span-4" runat="server"
ValidationGroup="RegisterValidationGroup"
            TextMode="Password" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxFirstNameLabel" AssociatedControlID="uxFirstName"
CssClass="span-3 last"
            runat="server" Text="FirstName :" />
        <ektronUI:TextField ID="uxFirstName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxLastNameLabel" AssociatedControlID="uxLastName"
CssClass="span-3 last"
            runat="server" Text="LastName :" />
        <ektronUI:TextField ID="uxLastName" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxDisplayLabel" AssociatedControlID="uxDisplay"
CssClass="span-3 last"
            runat="server" Text="Display :" />
        <ektronUI:TextField ID="uxDisplay" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
    <li class="clearfix">
        <ektronUI:Label ID="uxEmailLabel" AssociatedControlID="uxEmail" CssClass="span-3
last"
            runat="server" Text="Email :" />
        <ektronUI:TextField ID="uxEmail" CssClass="span-6" runat="server"
ValidationGroup="RegisterValidationGroup" />
    </li>
        <li class="clearfix" style="color: red;">
        <ektronUI:Label ID="uxStatus" Visible="false" CssClass="span-9 last"
runat="server"
            Text="Status:" />
    </li>
    <ektronUI:Button ID="uxSubmit" runat="server" OnClick="uxSubmit_Click"
Text="Update"></ektronUI:Button>
</ol>

<asp:Literal ID="uxMessage" runat="server"></asp:Literal>
```

.aspx.cs code-behind namespace

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using Ektron.Site;
using Ektron.Cms;
using Ektron.Cms.Framework.User;
```

.aspx.cs code-behind method

```
protected void uxSubmit_Click(object sender, EventArgs e)
    {
        try
        {
            UserManager Usermanager = new UserManager();
            long userID = long.Parse(uxUserId.Text);
            //Check the Valid EmailAddress or Not
            string emailAddress = uxEmail.Text;
            if (!string.IsNullOrEmpty(emailAddress))
            {
                try
                {
                    System.Net.Mail.MailAddress addr = new System.Net.Mail.MailAddress
(emailAddress);
                }
                catch
                {
                    uxStatus.Visible = true;
                    MessageUtilities.UpdateMessage(uxStatus, "Please Enter Valid Email
Address.", Message.DisplayModes.Error);
                    return;
                }
            }

            //Set the User Details
            UserData Userdata = Usermanager.GetItem(userID, true);
            if (Userdata != null)
            {
                Userdata.Username = uxUserName.Text != string.Empty ? uxUserName.Text :
Userdata.Username;
                Userdata.Password = uxPassword.Text != string.Empty ? uxPassword.Text :
Userdata.Password;
                Userdata.FirstName = uxFirstName.Text != string.Empty ? uxFirstName.Text
: Userdata.FirstName;
                Userdata.LastName = uxLastName.Text != string.Empty ? uxLastName.Text :
Userdata.LastName;
                Userdata.DisplayName = uxDisplay.Text != string.Empty ? uxDisplay.Text :
Userdata.DisplayName;
                Userdata.Email = uxEmail.Text != string.Empty ? uxEmail.Text :
Userdata.Email;

                Userdata.CustomProperties["Phone"].Value = "603-555-9999";
                Userdata.CustomProperties["Time Zone"].Value = "Eastern Standard Time";

                //Updates the UserDetails
                Usermanager.Update(Userdata);
                MessageUtilities.UpdateMessage(uxMessage, "User Details has been Updated
for UserID :" + Userdata.Id, Message.DisplayModes.Success);
```

```
            }
            else
            {
                MessageUtilities.UpdateMessage(uxMessage, "Please Enter Valid UserID",
Message.DisplayModes.Error);
            }
            uxPageMultiView.SetActiveView(uxViewMessage);
        }
        catch (Exception ex)
        {
            MessageUtilities.UpdateMessage(uxMessage, ex.Message,
Message.DisplayModes.Error);
            uxPageMultiView.SetActiveView(uxViewMessage);
        }

    }
```

Data Classes

UserCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.User
```

Properties

- `ReturnCustomeProperties`. Gets or sets the `ReturnCustomProperties` flag. If true, each user's custom properties also will be returned with results.

  ```
  public bool ReturnCustomProperties { set; get; }
  ```

- `UserCriteria()`

  ```
  public UserCriteria()
  ```

- `UserCriteria(Ektron.Cms.User.UserProperty, EkEnumeration.OrderByDirection)`

  ```
  public UserCriteria(Ektron.Cms.User.UserProperty
    orderByField, EkEnumeration.OrderByDirection orderByDirection)
  ```

  Accepted fields for the `UserProperty` are:

  - AcceptedTerms
  - Address
  - AddressName
  - AuthenticationTypeId
  - AuthenticationUserId
  - AuthGuid
  - Avatar
  - DateCreated

- DisplayName
- DisplayUserName
- Domain
- EditorOpen
- Email
- FirstName
- Id
- IsActivated
- IsDeleted
- IsDisableMessage
- IsMemberShip
- LanguageId
- LastLoginDate
- LastName
- Latitude
- LoginAttempts
- LoginIdentification
- Longitude
- Path
- Signature
- UserName
- VerifyGuid

UserCustomPropertyCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.User
```

Properties

- `AddFilter(string, Ektron.Cms.Common.CriteriaFilterOperator, object)`. Adds a Custom property based filter to the criteria object and returns the `UserCustomPropertyFilterGroup` that was created.

```
public Ektron.Cms.User.UserCustomPropertyFilterGroup
  AddFilter(string propertyName,
  Ektron.Cms.Common.CriteriaFilterOperator
  filterOperator, object value)
```

- `AddFilter(long, Ektron.Cms.Common.CriteriaFilterOperator, object)`. Adds a custom property based filter to the criteria object and returns the `UserCustomPropertyFilterGroup` that was created.

```
public Ektron.Cms.User.UserCustomPropertyFilterGroup
  AddFilter(long propertyId,
```

```
  Ektron.Cms.Common.CriteriaFilterOperator
  filterOperator, object value)
```

- `CustomPropertyFilterGroups`

```
public System.Collections.Generic.List<UserCustomPropertyFilterGroup>
  CustomPropertyFilterGroups { set; get; }
```

- `GenerateSql(System.Data.Common.DbCommand, System.Collections.Generic.Dictionary<Ektron.Cms.User.UserProperty,string>, out string, out string)`

```
public override void GenerateSql(System.Data.Common.DbCommand command,
  System.Collections.Generic.Dictionary<UserProperty,string>
  columnMap, out string whereClause, out string orderByClause)
```

- `ToCacheKey()`

```
public override string ToCacheKey()
```

- `UserCustomPropertyCriteria()`

```
public UserCustomPropertyCriteria()
```

- `UserCustomPropertyCriteria(Ektron.Cms.User.UserProperty, EkEnumeration.OrderByDirection)`

```
public UserCustomPropertyCriteria(Ektron.Cms.User.UserProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

Accepted fields for the `UserProperty` are:

- AcceptedTerms
- Address
- AddressName
- AuthenticationTypeId
- AuthenticationUserId
- AuthGuid
- Avatar
- DateCreated
- DisplayName
- DisplayUserName
- Domain
- EditorOpen
- Email
- FirstName
- Id
- IsActivated
- IsDeleted
- IsDisableMessage
- IsMemberShip
- LanguageId
- LastLoginDate

- LastName
- Latitude
- LoginAttempts
- LoginIdentification
- Longitude
- Path
- Signature
- UserName
- VerifyGuid

UserData

Namespace

```
Ektron.Cms
```

Properties

- `AcceptedTerms`. Gets or set whether the user has accepted the Terms and Conditions associated with a Threaded Discussion. This is for the UserData object. Returns boolean indicating whether the user has accepted the Terms and Conditions associated with a Threaded Discussion.

  ```
  public bool AcceptedTerms { set; get; }
  ```

- `Add(string)`. Adds a custom attribute by name

  ```
  public void Add(string sName)
  ```

- `AdUserName`. Obsolete: Gets or sets the Active Directory user name for the UserData object.

  ```
  public string AdUserName { set; get; }
  ```

- `AttributeList`. Gets custom attribute list as array

  ```
  public Ektron.Cms.CustomAttribute[] AttributeList { set; get; }
  ```

- `AuthenticationToken`. Gets or sets the authentication token assigned to this user for non-logged-in authentication purposes.

  ```
  public string AuthenticationToken { set; get; }
  ```

- `AuthenticationTokenExpires`. Gets or sets the datetime when the current authentication token expires.

  ```
  public System.DateTime AuthenticationTokenExpires { set; get; }
  ```

- `AuthenticationTypeId`. Gets or sets the ID of the authentication type this user uses to authenticate against for the CMS. For example, `EkEnumeration.UserAuthenticationType.Facebook` is used for integrated Facebook authentication.

  ```
  public long AuthenticationTypeId { set; get; }
  ```

- `AuthenticationUserId`. Gets or sets the id associated with this user in the third party authentication process. For Example, when `AuthenticationTypeId` is `EkEnumeration.UserAuthenticationType.Facebook`,

AuthenticationUserId would be the user's Facebook User ID.

```
public string AuthenticationUserId { set; get; }
```

- AuthGuid. Obsolete: When using a non CMS authentication method, the guid of the user in that system. Example: Active Directory objectGuid.

```
public string AuthGuid { set; get; }
```

- CustomAttribute(string). Gets a customAttribute instance by name

```
public Ektron.Cms.CustomAttribute CustomAttribute(string sName)
```

- CustomProperties. Gets the current users list of custom properties.

```
public Ektron.Cms.CustomAttributeList CustomProperties { set; get; }
```

- DateCreated. Gets or sets the creation date of a user. This is for the UserData object. Returns the date the user was created.

```
public string DateCreated { set; get; }
```

- DateModified. Gets the last modified date of a user. This is for the UserData object. Returns the date the user was last modified.

```
public System.DateTime DateModified { set; get; }
```

- EditorOption. Represents the content editor type configured on this user

```
public string EditorOption { set; get; }
```

- GetCustomAttributes(). Gets Custom attributes

```
public Ektron.Cms.CustomAttributeList GetCustomAtrributes()
```

- GetUserattributes(). Gets all custom attributes for this user and returns a list custom attributes.

```
public Ektron.Cms.CustomAttributeList GetUserAttributes()
```

- GroupId. Obsolete: To retrieve a list of groups this User belongs to use Ektron.Cms.API.User.GetAllUserGroups() function. Gets or sets the numeruc ID (Long) of the group to which the user belongs. This is for the UserData object. Returns the group ID to which the user belongs.

```
public long GroupId { set; get; }
```

- GroupName. Obsolete: To retrieve a list of groups this User belongs to use Ektron.Cms.API.User.GetAllUserGroups() function. Gets or sets the numeric ID (Long) of the group to which the user belongs. This is for the UserData object. Returns the name of the group to which the user belongs.

```
public string GroupName { set; get; }
```

- IsAccountLocked(EkRequestInformation). Gets a flag that indicates whether this account is locked or not and returns true when account is locked; false when not.

```
public bool IsAccountLocked(EkRequestInformation reqInfo)
```

- IsDeleted. Gets or sets a boolean flag that indicates if the user was deleted.

```
public bool IsDeleted { set; get; }
```

- IsDisableMessage. Gets or sets a flag that indicates whether messaging is enabled or disabled for this user.

```
public bool IsDisableMessage { set; get; }
```

- `IsMemberShip`. Gets or sets a boolean indicating whether the user is a Membership user or not. This is for UserData object. Returns a boolean indicating whether the user is a Membership user.

```
public bool IsMemberShip { set; get; }
```

- `LanguageId`. Gets or sets the integer ID of the language for the UserData object and returns the integer ID of the language.

```
public int LanguageId { set; get; }
```

- `LanguageName`. Gets the name of the user's language for the UserData object and returns the name of the user's language.

```
public string LanguageName { set; get; }
```

- `LastLoginDate`. Gets or sets the last date a user logged into Ektron CMS. This is for the UserData object. Returns the last login date for a user.

```
public string LastLoginDate { set; get; }
```

- `LastPasswordChangeDate`. Gets or sets the last date a user changed their password. Returns the date when the user last changed the password.

```
public System.DateTime LastPasswordChangeDate { set; get; }
```

- `LoginAttempts`. Gets an Int16 representing the number of times a user has attempted to login for the UserData object. This number resets to 0 (zero) once a user successfully logs in.

```
public short LoginAttempts { set
```

- `LoginIdentification`. Gets or sets a unique login ID

```
public string LoginIdentification { set; get; }
```

- `PasswordExpired(int)`. Gets a boolean representing if the user's password would expire given a certain amount of days from today's date and returns true when password expired; false otherwise.

```
public bool PasswordExpired(int expirationPeriodInDays)
```

- `Path`. Obsolete.

```
public string Path { set; get; }
```

- `Rank`. Gets or sets the user's rank for a Threaded Discussion for the UserData object and returns the user's rank for a Threaded Discussion.

```
public Ektron.Cms.UserRank Rank { set; get; }
```

- `SessionId`. Gets the users current session Id.

```
public System.Guid SessionId { set; get; }
```

- `Signature`. Gets or sets the user's signature for the UserData object, used in the forums and other locations.

```
public string Signature { set; get; }
```

- `UserPreference`. Gets or sets a user's preferences as UserPreferenceData for the UserData object.

```
public Ektron.Cms.UserPreferenceData UserPreference { set; get; }
```

- `Validate()`. Validate this instance's data.

```
public override ValidationResults Validate()
```

UserTaxonomyCriteria

See also *Criteria use for GetList methods* on page 156 for more information.

Namespace

```
Ektron.Cms.User
```

Properties

- `AddFilter(string, [bool])`

```
public Ektron.Cms.User.UserTaxonomyFilterGroup
  AddFilter(string taxonomyPath, [bool recursive = false])
```

- `AddFilter(long, [bool])`

```
public Ektron.Cms.User.UserTaxonomyFilterGroup
  AddFilter(long taxonomyId, [bool recursive = false])
```

- `AddFilter(Ektron.Cms.User.UserTaxonomyProperty, Ektron.Cms.Common.CriteriaFilterOperator, object, [bool])`

```
public Ektron.Cms.User.UserTaxonomyFilterGroup
  AddFilter(Ektron.Cms.User.UserTaxonomyProperty
  taxonomyProperty, Ektron.Cms.Common.CriteriaFilterOperator
  filterOperator, object value, [bool recursive = false])
```

- `GenerateSql(System.Data.Common.DbCommand, System.Collections.Generic.Dictionary<Ektron.Cms.User.UserProperty,string>, out string, out string)`

```
public override void GenerateSql(System.Data.Common.DbCommand command,
  System.Collections.Generic.Dictionary<UserProperty,string>
  columnMap, out string whereClause, out string orderByClause)
```

- `ToCacheKey()`

```
public override string ToCacheKey()
```

- `UserTaxonomyCriteria()`

```
public UserTaxonomyCriteria()
```

- `UserTaxonomyCriteria(Ektron.Cms.User.UserProperty, EkEnumeration.OrderByDirection)`

```
public UserTaxonomyCriteria(Ektron.Cms.User.UserProperty
  orderByField, EkEnumeration.OrderByDirection orderByDirection)
```

  Accepted fields for the `UserProperty` are:

  - AcceptedTerms
  - Address
  - AddressName
  - AuthenticationTypeId
  - AuthenticationUserId
  - AuthGuid
  - Avatar
  - DateCreated

- DisplayName
- DisplayUserName
- Domain
- EditorOpen
- Email
- FirstName
- Id
- IsActivated
- IsDeleted
- IsDisableMessage
- IsMemberShip
- LanguageId
- LastLoginDate
- LastName
- Latitude
- LoginAttempts
- LoginIdentification
- Longitude
- Path
- Signature
- UserName
- VerifyGuid

2

Framework UI

8.50 and higher

The framework user interface (UI) lets you interact with the following Ektron CMS user controls.

- Accordion. Provides vertically tabbed headers to display multiple sections of content.
- Autocomplete. Provides choices in a text field from a pre-populated list of values.
- Blueprint. Implements style standards to define consistent column and line height styles.
- Button. Creates buttons with an input of type `submit`, `reset`, or `anchors` are themable buttons with mouseover and active styles.
- ButtonSet. Groups buttons with their own naming container.
- Css. Lets you associate a Cascading Style Sheet (CSS) to an ASP.NET control.
- CssBlock. Lets you define and apply inline Cascading Style Sheet (CSS) rules inside any ASP.NET control without authoring invalid HTML markup.
- DateField. Creates an input field that allows only a date value that is in a culturally sensitive format.
- DatePicker. Lets you double click to post a date to your page.
- DecimalField. Creates an input field that allows only decimal values.
- Dialog. Creates a floating window that contains a title bar, content area, and dialog buttons.
- InfoTip. 9.00 and higher Initially hides a message and makes it visible "on demand."
- IntegerField. Creates an input field that allows only integers.
- JavaScript. Associates an external file containing JavaScript (JS) code to an ASP.NET control.
- JavaScriptBlock. Lets you define and apply inline JavaScript (JS) rules inside any ASP.NET control without authoring invalid HTML markup.
- Label. Outputs an HTML `<label>` tag that is different than the tag in a standard ASP label control.
- Message. Displays messages to users of an application in a consistent and themable format.
- Pager. Limits the number of items on a page and provide simple navigation to other items.
- Tabs. Provides horizontally tabbed headers to display multiple sections of content.
- TextField. Creates and input field in which you can enter text.
- Tree. 9.00 and higher Renders items that are added declaratively or programatically as a nested unordered-list.

- ValidationMessage. Displays the messages that are set in the associated server control's `ErrorMessage` property.
- Wizard. 9.00 and higher Simplifies the process of creating multi-step, branching user interfaces.

Accordion

8.50 and higher

```
<EktronUI:Accordion>
```

Provides vertically tabbed headers to display multiple sections of content. (See also *Tabs* on page 1979.)

When you click on an accordion tab, the other tabs slide up or down to reveal the content of the selected tab.

Events for Accordion

- `Click`. Set to server side event handler that is called when a user clicks the tab. Must be set on the contained Tab control.
- `Command`. The Command event is raised when the control is clicked. By associating a command name with the control, one method can handle the event from multiple controls, and you can programmatically determine the specific control that triggered the event by examining the event's command name.

Methods for Accordion

- `SetActiveTab(EktronUI.Tab)`. Selects the supplied tab; must be a child of the `<EktronUI:Tabs>` control.
- `SetActiveTab(string)`. Selects the child tab that has the supplied ID.

Theming for Accordion

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div id="uxDemoAccordion" class="ektron-ui-control ektron-ui-accordion">
  <div role="tablist" class="ektron-ui-accordion-selectorui-accordion
    ui-widget ui-helper-reset ui-accordion-icons">
    <h3 tabindex="0" aria-selected="true" aria-expanded="true" role="tab"
      id="uxDemoAccordion_uxTab1_AccordionTemplate_aspAccordionHeader"
      class="EktronUI ui-accordion-headerui-helper-reset ui-state-default
      ui-state-active ui-corner-top">
      <span class="ui-icon ui-icon-triangle-1-s"></span>
      <a tabindex="-1" href="#uxDemoAccordion_uxTab1" title="Tab One">
        Tab One
```

```
      </a>
    </h3>
    <div role="tabpanel"
      class="ui-accordion-content ui-helper-reset ui-widget-content
        ui-corner-bottom ui-accordion-content-active">
      <h4>Inside tab one</h4>
        Lorem ipsum dolor sit amet, consectetur adipisicing elit,
        sed do eiusmod tempor incididunt ut labore …
      </div>
    ..
  </div>
</div>
```

Examples for Accordion

Accordion has the following variations:

- **Default Functionality**. Vertical tab controls display grouped content by expanding and collapsing.
- **Active Tab**. Shows the active tab that is set in the code-behind in the page's OnLoad method.
- **Contains Controls**. Clicking on a tab fills a field with postback text. AutoHeight is disabled, which causes each tab to fit their contents rather than adjusting all tabs to the size of the largest.
- **Databind to Add Tabs**. Adds a tab by databinding.
- **Dynamically Add Tabs**. Adds a tab programatically.
- **Postback**. Displays text that is associated with the click event. The Tabs control contains a server-side onclick event handler that generates a postback when clicked.

Default functionality example

.aspx

```
<ektronUI:Accordion ID="uxDemoAccordion" runat="server" >
    <ektronUI:Tab ID="uxTab1" runat="server" CssClass="EktronUI" Text="Tab One" >
        <ContentTemplate>
            <h2>Inside tab one</h2>
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
               sed do eiusmod tempor incididunt ut labore et dolore
               magna aliqua.</p>
        </ContentTemplate>
    </ektronUI:Tab>
    <ektronUI:Tab ID="uxTab2" runat="server" CssClass="EktronUI" Text="Tab Two" >
        <ContentTemplate>
            <h2>Inside tab two</h2>
            <p>Ut enim ad minim veniam, quis nostrud exercitation
               ullamco laboris nisi ut aliquip ex ea commodo consequat.
               Duis aute irure dolor in reprehenderit in voluptate
               velit esse cillum dolore eu fugiat nulla pariatur.</p>
        </ContentTemplate>
    </ektronUI:Tab>
```

```
    <ektronUI:Tab ID="uxTab3" runat="server" CssClass="EktronUI"
      Text="Tab Three" >
        <ContentTemplate>
            <h2>Inside tab three</h2>
            <p>Excepteur sint occaecat cupidatat non proident,
               sunt in culpa qui officia deserunt mollit anim id
               est laborum.</p>
        </ContentTemplate>
    </ektronUI:Tab>
</ektronUI:Accordion>
```

NOTE: .aspx.cs not required.

Active Tab example

.aspx

```
<ektronUI:Accordion ID="uxDemoAccordion" runat="server" >
    <ektronUI:Tab ID="uxTab1" runat="server" CssClass="EktronUI" Text="Tab One" >
        <ContentTemplate>
            <h2>Inside tab one</h2>
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
               sed do eiusmod tempor incididunt ut labore et dolore
               magna aliqua.</p>
        </ContentTemplate>
    </ektronUI:Tab>
    <ektronUI:Tab ID="uxTab2" runat="server" CssClass="EktronUI" Text="Tab Two" >
        <ContentTemplate>
            <h2>Inside tab two</h2>
            <p>Ut enim ad minim veniam, quis nostrud exercitation
               ullamco laboris nisi ut aliquip ex ea commodo consequat.
               Duis aute irure dolor in reprehenderit in voluptate
               velit esse cillum dolore eu fugiat nulla pariatur.</p>
        </ContentTemplate>
    </ektronUI:Tab>
    <ektronUI:Tab ID="uxTab3" runat="server" CssClass="EktronUI" Text="Tab Three" >
        <ContentTemplate>
            <h2>Inside tab three</h2>
            <p>Excepteur sint occaecat cupidatat non proident,
               sunt in culpa qui officia deserunt mollit anim
               id est laborum.</p>
        </ContentTemplate>
    </ektronUI:Tab>
</ektronUI:Accordion>
```

.aspx.cs

```
protected override void OnLoad(EventArgs e)
{
    base.OnLoad(e);
    uxDemoAccordion.SetActiveTab(uxTab2);
}
```

Contains Controls example

.aspx

```
<ektronUI:Accordion ID="uxDemoAccordion" runat="server" >
    <ektronUI:Tab ID="uxTab1" runat="server" CssClass="EktronUI" Text="Tab One" >
        <ContentTemplate>
            <ektronUI:TextField ID="uxTextField" runat="server" Text="" />
            <ektronUI:ButtonSet ID="ButtonSet1" runat="server">
                <ektronUI:Button ID="Button1" runat="server"
                  Text="One" DisplayMode="Button" OnClick="Button1_click" />
                <ektronUI:Button ID="Button2" runat="server"
                  Text="Two" DisplayMode="Button" OnClick="Button2_click" />
                <ektronUI:Button ID="Button3" runat="server"
                  Text="Three" DisplayMode="Button" OnClick="Button3_click" />
            </ektronUI:ButtonSet>
        </ContentTemplate>
    </ektronUI:Tab>
    <ektronUI:Tab ID="uxTab2" runat="server" CssClass="EktronUI" Text="Tab Two" >
        <ContentTemplate>
            <cms:Login ID="login1" runat="server" /><br /><br />
            <cms:ContentBlock ID="cb1" runat="server" DefaultContentID="30" />
            <cms:ListSummary ID="ls1" runat="server" FolderID="78" Recursive="false" />
        </ContentTemplate>
    </ektronUI:Tab>
</ektronUI:Accordion>
```

.aspx.cs

```
protected void Button1_click(object sender, EventArgs e)
{
    uxTextField.Text = "Button one clicked!";
}

protected void Button2_click(object sender, EventArgs e)
{
    uxTextField.Text = "Button two clicked!";
}

protected void Button3_click(object sender, EventArgs e)
{
    uxTextField.Text = "Button three clicked!";
}
```

Databind to Add Tabs example

.aspx

```
<ektronUI:Accordion ID="uxAccordionControl"
   runat="server" CssClass="class_tabs1" ViewStateMode="Enabled" >
        <ektronUI:Tab ID="Tab0" runat="server" Text="Tab 0"
          CssClass="class_tab0">
            <ContentTemplate>
                Tab-Zero (declaratively added)
            </ContentTemplate>
```

```
        </ektronUI:Tab>
    </ektronUI:Accordion>
```

.aspx.cs

```
protected void Page_Init(object sender, EventArgs e) {
        this.SelectedTab = "uxEktronUiTab";
        this.SelectedAccordion = "uxAccordionTab";
        this.SelectedSample = "DatabindTab";
        this.ComponentName = "ektronUI:Accordion";
        this.ComponentUrl =
"/Framework/UI/Controls/EktronUI/Accordion/DatabindTab.aspx.aspx";
        this.IsDemoSafe = true;
        this.DocumentationName = "Ektron.Cms.Framework.UI.Controls.EktronUI.Tabs";
        uxSummary.BaseClasses.Add
("Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets.TabsBase");

        LoadTabControls();
    }

    protected void LoadTabControls() {

        var tabs = new TabData[]{
            new TabData(){Text = "Tab 1",
              ContentText="tab one (added by DataBinding)",
              Selected = true},
            new TabData(){Text = "Tab 2",
              ContentText="tab two (added by DataBinding)"},
            new TabData(){Text = "Tab 3",
              ContentText="tab three (added by DataBinding)"}
        };

        uxAccordionControl.DataSource = tabs;
        uxAccordionControl.DataBind();
    }
```

Dynamically Add Tabs example

.aspx

```
<ektronUI:Accordion ID="uxAccordionControl" runat="server" CssClass="class_tabs1">
            <ektronUI:Tab ID="Tab1" runat="server" Text="Tab1" >
                <ContentTemplate>
                    Tab1 (declaratively added)
                </ContentTemplate>
            </ektronUI:Tab>
            <ektronUI:Tab ID="Tab2" runat="server" Text="Tab2" >
                <ContentTemplate>
                    Tab2 (declaratively added)
                </ContentTemplate>
            </ektronUI:Tab>
        </ektronUI:Accordion>
        <br />
    <asp:Literal runat="server" ID="messageLiteral" />
```

.aspx.cs

```
protected void Page_Init(object sender, EventArgs e) {
        this.SelectedTab = "uxEktronUiTab";
        this.SelectedAccordion = "uxAccordionTab";
        this.SelectedSample = "DynamicAddTab";
        this.ComponentName = "ektronUI:Accordion";
        this.ComponentUrl =
"/Framework/UI/Controls/EktronUI/Accordion/DynamicAddTab.aspx";
        this.IsDemoSafe = true;
        this.DocumentationName = "Ektron.Cms.Framework.UI.Controls.EktronUI.Tabs";
        uxSummary.BaseClasses.Add
("Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets.TabsBase");

        LoadTabControls();
    }

    protected void LoadTabControls() {
        var tab = new Tab { Text = "Dynamic Tab1", ID = "DynamicTab1",
          Content = "Inside Dynamic Tab1 (added programatically)"};
        uxAccordionControl.AddTab(tab);

        tab = new Tab {Text = "Dynamic Tab2", ID = "DynamicTab2",
          Content = "Inside Dynamic Tab2 (added programatically)"};
        uxAccordionControl.AddTab(tab);
    }
```

Postback example

.aspx

```
<ektronUI:Accordion ID="uxDemoAccordion" runat="server" >
    <ektronUI:Tab ID="uxTab1" runat="server" CssClass="EktronUI"
      Text="Tab One" >
        <ContentTemplate>
            Static content... (click the other tab)
        </ContentTemplate>
    </ektronUI:Tab>
    <ektronUI:Tab ID="uxTab2" runat="server" CssClass="EktronUI"
      Text="Tab Two" OnClick="uxTab2_click" >
        <ContentTemplate>
            <b>
                <asp:Literal ID="literal1" runat="server" />
            </b>
        </ContentTemplate>
    </ektronUI:Tab>
</ektronUI:Accordion>
```

.aspx.cs

```
protected void uxTab2_click(object sender, EventArgs e)
{
    uxTab2.Text = "Tab Two (been clicked)";
    uxDemoAccordion.SetActiveTab(uxTab2);
```

```
    literal1.Text = "Content set in code-behind, during click event servicing...)";
}
```

AdaptiveMultiView

9.00 and higher

```
<EktronUI:AdaptiveMultiView>
```

Lets you specify sections of a template that will appear for a specific Device Group. When a page is loaded, the template determines which device group the visitor's device fits into and displays the appropriate view for this control.

This UI control associates each "AdaptiveView" section to one of the Device Groups created in the Workarea (**Workarea > Settings > Configuration > Mobile Settings > View All Device Groups**).

AdaptiveMultiView syntax:

```
<ektronUI:AdaptiveMultiView runat="server" ID="MyMobileControl">
<ektronUI:AdaptiveView ID="AdaptiveView1" runat="server" GroupId="7"
  GroupType="Breakpoint">
    I'm a SmartPhone!
</ektronUI:AdaptiveView>
<ektronUI:AdaptiveView ID="AdaptiveView2" runat="server" GroupId="3"
  GroupType="DeviceGroup">
    I'm a CoolPhone!
</ektronUI:AdaptiveView>
<ektronUI:AdaptiveView ID="AdaptiveView3" runat="server" GroupId="0"
  GroupType="DeviceGroup">
    I'm Generic Mobile!
</ektronUI:AdaptiveView>
<ektronUI:AdaptiveView ID="AdaptiveView4" runat="server" IsDefaultView="true">
    I'm a Default!
</ektronUI:AdaptiveView>
</ektronUI:AdaptiveMultiView>
```

AdaptiveMultiView properties

- **GroupId** (Long)

  Group Id for the Device Group, which is displayed you can find in the table at **Workarea > Settings > Configuration > Mobile Settings > View All Device Groups**. If you set this to 0, GroupID is associated with any device that does not fit into any of the Device Groups created.

  NOTE: If Breakpoint Groups exist, every device will fall into one of these groups, therefore making this "Generic Mobile" group unusable.

- **GroupType** (String)

  Determines whether the group is a Breakpoint Group or a manually created Device Configuration. The possible values for this property are `Breakpoint` or `DeviceGroup`.

- **IsDefaultView** (Boolean)
  Set this value to "true" to display this view for any visitors using a device that doesn't fit into any other group. This view will typically be displayed for any desktop visitors.

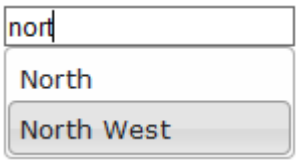
Autocomplete

8.50 and higher

```
<EktronUI:Autocomplete>
```

Provides choices in a text field from a pre-populated list of values. For example, if the field asks for a *State* name in the USA and the user types in `new`, all the states starting with *New* such as *New Hampshire*, *New Jersey*, *New Mexico*, and *New York* appear in a list where a user can use the cursor or mouse to select any of the suggestions. You can narrow the options by entering more characters.

The following example finds values with `nort` in them.



Templates

- SourceData
- ValidationRules

Validation Rules

- CustomRule
- RegexRule
- RequiredRule

Events for Autocomplete

None.

Methods for Autocomplete

None.

Theming for Autocomplete

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<span class="ektron-ui-control ektron-ui-autocomplete"
  id="bluePrint_autocomplete">
  <span class="ektron-ui-control ektron-ui-input ektron-ui-textField"
   id="bluePrint_autocomplete_Autocomplete_uxAutocompleteTextBox_
     TextField_uxTextField">
  <input type="text"
   id="bluePrint_autocomplete_Autocomplete_uxAutocompleteTextBox_
     TextField_aspInput"
   name="bluePrint$autocomplete$Autocomplete$uxAutocompleteTextBox
     $TextField$aspInput">
  </span>
</span>
```

Example for Autocomplete

Default functionality example

.aspx

```
<ektronUI:Label runat="server" ID="uxLabel" AssociatedControlID="uxAutoComplete"
Text="Region:" />
<ektronUI:Autocomplete runat="server" ID="uxAutocomplete">
    <SourceData>
        <ektronUI:AutocompleteData Label="North" Value="North" />
        <ektronUI:AutocompleteData Label="South" Value="South" />
        <ektronUI:AutocompleteData Label="East" Value="East" />
        <ektronUI:AutocompleteData Label="West" Value="West" />
        <ektronUI:AutocompleteData Label="North West" Value="North West" />
    </SourceData>
</ektronUI:Autocomplete>
```

NOTE: .aspx.cs not required.

Blueprint

8.50 and higher

```
<EktronUI:Blueprint>
```

Implements style standards to define consistent column and line height styles.

- Provides an easy to use framework for consistent HTML element placements
- Improves UI consistency
- Reduces CSS coding errors
- Eliminates need for HTML tables for layouts
- Provides easy cross-browser support
- Reduces implementation time for projects
- Built-in styles for print and screen

You can use `<EktronUI:Blueprint>` in PageBuilder page layouts. However, you should not use it inside widgets.

From www.blueprintcss.org: "*Blueprint is a CSS framework, which aims to cut down on your development time. It gives you a solid foundation to build your project on top of, with an easy-to-use grid, sensible typography, useful plugins, and even a stylesheet for printing.*"

`<EktronUI:Blueprint>` provides additional functionality to Blueprint classes.

- .NET based server control coding eases developer implementation
- Visual Studio IntelliSense provides prompts for options and classes already defined in Blueprint

`<EktronUI:Blueprint>` by itself is not visible in a browser but it does effect the styles of other elements and controls. Blueprint defines 24 columns of 30px each with a 10 px gutter between columns.

For example, on the Ektron OnTrek home page, under the banner there are three columns. Each column has identical width. Use a class="span-x" tag like this for each column instead of using a CSS width property like the following code snippets.

```
<div class="span-8"></div>
<div class="span-8"></div>
<div class="span-8 last"></div>
```

The class called *span-8* means that this element is contained in 8 of the 24 columns defined by `<EktronUI:Blueprint>`. An additional class called *last* is added to the right side column.

`<EktronUI:Blueprint>` also defines typography standards so that text and other UI objects like text boxes or buttons line up across columns. With proper style classes from `<EktronUI:Blueprint>`, you can align various tags such as `<h1>`, `<h4>`, `<p>` in a grid. (See http://blueprintcss.org/tests/parts/grid.html for an example of the Blueprint Grid.)

Events for Blueprint

None.

Methods for Blueprint

None.

Theming for Blueprint

`<EktronUI:Blueprint>` control uses Blueprint CSS Framework. For further deeper information on Blueprint CSS Framework, visit http://www.blueprintcss.org/.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div class="ektron-ui-control
            ektron-ui-blueprint
            container"
   id="bluePrintLayout_uxBodyColumn_phBody_phBody_
      ucbodyColumn_BlueprintColumn1_phDemo_blueprint1">
   <div class="ektron-ui-control
               ektron-ui-blueprintColumn
               span-17 bluePrintColumnClass"
    id="bluePrintLayout_uxBodyColumn_phBody_phBody_
         ucbodyColumn_BlueprintColumn1_phDemo_
         blueprint1_BlueprintColumn_Header">
      <p>HEADER</p>
   </div>
   <div class="ektron-ui-control
               ektron-ui-blueprintColumn
               span-6 colborder"
    id="bluePrintLayout_uxBodyColumn_phBody_phBody_
         ucbodyColumn_BlueprintColumn1_phDemo_
         blueprint1_BlueprintColumn_Body1">
      <div>
         <p>BODY COLUMN 1</p>
         <p>
         Etiam et purus vel dolor interdum venenatis
         nec sit amet tortor. Aliquam aliquam
         vestibulum urna a vulputate. Proin eget
         aliquam augue. Pellentesque ultrices justo
         eu velit mollis ut lobortis ipsum mattis.
         Nulla euismod nisl sed elit ultrices ornare
         et quis est.</p>
      </div>
   </div>
   <div class="ektron-ui-control
               ektron-ui-blueprintColumn
               last span-4"
    id="bluePrintLayout_uxBodyColumn_phBody_phBody_
         ucbodyColumn_BlueprintColumn1_phDemo_
```

```
         blueprint1_BlueprintColumn_Body2">
      <div>
         <p>BODY COLUMN 2</p>
         <p>
         Etiam et purus vel dolor interdum venenatis
         nec sit amet tortor. Aliquam aliquam
         vestibulum urna a vulputate. Proin eget
         aliquam augue. Pellentesque ultrices justo
         eu velit mollis ut lobortis ipsum mattis.
         Nulla euismod nisl sed elit ultrices ornare
         et quis est.</p>
      </div>
   </div>
   <div class="ektron-ui-control
               ektron-ui-blueprintColumn
               span-17 bluePrintColumnClass"
      id="bluePrintLayout_uxBodyColumn_phBody_phBody_
         ucbodyColumn_BlueprintColumn1_phDemo_
         blueprint1_BlueprintColumn_Footer">
      <p>FOOTER</p>
   </div>
</div>
```

Example for Blueprint

Default functionality example

.aspx

```
<ektronUI:CssBlock ID="uxCssBlock" runat="server">
        <CssTemplate>
            .bluePrintColumnClass { border: 1px none silver;}
        </CssTemplate>
</ektronUI:CssBlock>
<ektronUI:Blueprint ID="blueprint1" runat="server">
    <ektronUI:BlueprintColumn ID="BlueprintColumn_Header" runat="server" Span="17"
CssClass="bluePrintColumnClass">
        <p>HEADER</p>
    </ektronUI:BlueprintColumn>
    <ektronUI:BlueprintColumn ID="BlueprintColumn_Body1" runat="server" Span="6"
ColumnBorder="true">
        <div>
            <p>BODY COLUMN 1</p>
            <p>
                Etiam et purus vel dolor interdum venenatis nec sit amet tortor. Aliquam
aliquam
                vestibulum urna a vulputate. Proin eget aliquam augue. Pellentesque
ultrices justo
                eu velit mollis ut lobortis ipsum mattis. Nulla euismod nisl sed elit
ultrices ornare
                et quis est.</p>
        </div>
    </ektronUI:BlueprintColumn>
    <ektronUI:BlueprintColumn ID="BlueprintColumn_Body2" runat="server" Span="4"
IsLast="true">
```

```
        <div>
            <p>BODY COLUMN 2</p>
            <p>
                Etiam et purus vel dolor interdum venenatis nec sit amet tortor. Aliquam
aliquam
                vestibulum urna a vulputate. Proin eget aliquam augue. Pellentesque
ultrices justo
                eu velit mollis ut lobortis ipsum mattis. Nulla euismod nisl sed elit
ultrices ornare
                et quis est.
            </p>
        </div>
    </ektronUI:BlueprintColumn>
    <ektronUI:BlueprintColumn ID="BlueprintColumn_Footer" runat="server" Span="17"
CssClass="bluePrintColumnClass">
        <p>FOOTER</p>
    </ektronUI:BlueprintColumn>
</ektronUI:Blueprint>
```

NOTE: .aspx.cs not required.

Button

8.50 and higher

```
<EktronUI:Button>
```

Creates buttons with an input of type `submit`, `reset`, or `anchors` are buttons that can be themed with mouseover and active styles. When you use an input of type `button`, `submit` or `reset`, the buttons can have only plain text labels with no icons.

Properties

- `CheckableControlClientID`. Sets the container control's Clientid as the underlying ASP check box or ASP radio button's ID if the display mode of the button control is either radio button or check box.
- `CheckableControlUniqueID`. Sets the container control's Uniqueid as the underlying ASP check box or ASP radio button's ID if the display mode of the button control is either radio button or check box.
- `Checked`. Gets of sets the boolean value for the check box and radio button.
- `ClientEvents`. Gets or sets the client events property for the control.
- `DisplayMode`. Gets or sets the display type of the following input types check box, radio button, submit, and button.
- `Enabled`. Gets or sets a value indicating whether the control can respond to user interaction. Default value is true.
- `IsButtonSet`. Gets a boolean value for whether the current button is part of a button set.
- `Options`. Gets or sets the available options for the control. Options for this control are Disabled, HasJson, Icon, Label, and Text.
- `PrimaryIcon`. Gets or sets the UI ThemeRoller for Primary Icon for a given button, which is displayed left of the text. Does not apply to type button and submit.
- `SecondaryIcon`. Gets or sets the UI ThemeRoller Secondary Icon for a given button, which is displayed right of the text. Does not apply to type button and submit.
- `Selector`. Gets or sets the hierarchical id of the control which can be used in jQuery as a selector.

Events for Button

- `create`. When button is created, this event is triggered.

Methods for Button

None.

Theming for Button

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

Sample markup with Ektron UI CSS Framework classes:

```
<span class="ektron-ui-control ektron-ui-button" id="bluePrint_button">
   <button title="" onclick="" id="bluePrint_button_Button"
     class="ui-button ui-widget ui-state-default ui-corner-all
       ui-button-text-only" role="button">
     <span class="ui-button-text">ButtonText</span>
   </button>
</span>
```

Examples for Button

Button has the following variations:

- **Default Functionality**. Assigned a standard form input field of type button. Use standard ASP Button events to generate client-side or postback events.
- **Check Box**. Styled as a highlighting toggle button with the button widget. The button text comes from the label element associated with the check box.
- **Icons**. Metadata specifies a combination of icon and text.
- **Radio Button**. Shows a set of buttons where only one can be selected (highlighted) at a time.
- **Submit**. A postback of the form is sent for server-side events.
- **Toolbar**. Specifies a mix of check box and radio buttons (for example, for a media player toolbar). See the examples in this section.
- **Update Panel**. The page does a partial post back and the page fields are updated with the values added in text box.

Default functionality example

.aspx

```
<ektronUI:Button ID="uxButton" DisplayMode="Button"
  Text="A Button Element"
  OnClientClick="alert('Button enhances standard form
    elements like button, input of type submit, anchor,
    radio button and checkbox that is themable with
```

```
    appropriate mouseover and active styles.');"
  runat="server" />
```

NOTE: .aspx.cs not required.

Check Box example

.aspx

```
<ektronUI:Button ID="uxCheckBox1" DisplayMode="Checkbox"" runat="server" Text="Option1"
/>
<ektronUI:Button ID="uxCheckBox2" DisplayMode="Checkbox"" runat="server" Text="Option2"
/>
<ektronUI:Button ID="uxCheckBox3" DisplayMode="Checkbox"" runat="server" Text="Option3"
/>
```

NOTE: .aspx.cs not required.

Icons example

.aspx

```
<ektronUI:Button ID="uxIcon" PrimaryIcon="Alert" Text="Icon" runat="server" / >
<ektronUI:Button ID="uxPrimaryIcon" PrimaryIcon="ArrowDiag" Text="Primary Icon"
runat="server" />
<ektronUI:Button ID="uxSecondaryIcon" SecondaryIcon="Alert" Text="Secondary Icon"
runat="server" />
<ektronUI:Button ID="uxPrimarySecondaryIcon" SecondaryIcon="Alert" Text="Primary -
Secondary Icon" runat="server" />
```

NOTE: .aspx.cs not required.

Radio Button example

.aspx

```
<ektronUI:Button ID="uxCheckBox1" DisplayMode="RadioButton"" runat="server"
Text="Option1" >
<ektronUI:Button ID="uxCheckBox2" DisplayMode="RadioButton"" runat="server"
Text="Option2" >
<ektronUI:Button ID="uxCheckBox3" DisplayMode="RadioButton"" runat="server"
Text="Option3" >
```

NOTE: .aspx.cs not required.

Submit example

.aspx

```
<ektronUI:Button ID="uxSubmit" DisplayMode="Submit"" runat="server" Text="Option1" >
<ektronUI:Label ID="uxLabel" unat="server" />
```

NOTE: .aspx.cs not required.

Toolbar example

.aspx

```
<div style="padding: 5px;">
    <span style="padding: 10px 4px;" class="ui-widget-header ui-corner-all">
        <ektronUI:Button ID="uxSeekFirst" DisplayMode="Anchor" Text=""
PrimaryIcon="SeekFirst" runat="server" />
        <ektronUI:Button ID="uxSeekPrevious" DisplayMode="Anchor" Text=""
PrimaryIcon="SeekPrevious" runat="server" />
        <ektronUI:Button ID="uxPlay" DisplayMode="Anchor" Text="" PrimaryIcon="Play"
runat="server" />
        <ektronUI:Button ID="uxStop" DisplayMode="Anchor" Text="" PrimaryIcon="Stop"
runat="server" />
        <ektronUI:Button ID="uxSeekNext" DisplayMode="Anchor" Text=""
PrimaryIcon="SeekNext" runat="server" />
        <ektronUI:Button ID="uxSeemEnd" DisplayMode="Anchor" Text=""
PrimaryIcon="SeekEnd" runat="server" />
    </span>
</div>
```

NOTE: .aspx.cs not required.

Update Panel example

.aspx

```
<ektronUI:CssBlock runat="server" ID="uxCSSBlock">
    <CssTemplate> .listStyle {list-style:none;} </CssTemplate>
</ektronUI:CssBlock>
<ul>
    <li class="listStyle">First Name:
        <asp:TextBox runat="server" ID="aspFirstName"></asp:TextBox>
    </li>
    <li class="listStyle">Last Name:
        <asp:TextBox runat="server" ID="aspLastName"></asp:TextBox>
    </li>
    <li class="listStyle">
        <asp:ScriptManager runat="server" ID="ScriptManager1">
        </asp:ScriptManager>
        <asp:UpdatePanel runat="server" ID="aspUpdatePanel">
            <ContentTemplate>
                <ektronUI:Button runat="server" ID="uxButton" OnClick="uxButton_Click"
    Text="Update Panel Demo">>/ektronUI:Button>
                The First Name is <b>
                    <asp:Label runat="server" ID="lblFirstName"></asp:Label></b>
    and the last name is <b>
                    <asp:Label runat="server" ID="lblLastName"></asp:Label></b>
            </ContentTemplate>
        </asp:UpdatePanel>
    </li>
</ul>
```

.aspx.cs

```
protected void uxButton_Click(object sender, EventArgs e)
{
    lblFirstName.Text = "\"" + aspFirstName.Text +"\"";
    lblLastName.Text = "\"" + aspLastName.Text + "\".";
}
```

ButtonSet

8.50 and higher

```
<EktronUI:ButtonSet>
```

Groups buttons with their own naming container. For example, placing a set of radio buttons inside a `<EktronUI:ButtonSet>` control lets the radio buttons interact within their own group of radio buttons and ignore the behavior of all other radio buttons that might be placed on the template.

You can choose multiple options with a check box or a single option with a radio button.

See also *Button* on page 1922.

See also the following link for more information about *naming containers*.
http://msdn.microsoft.com/en-us/library/system.web.ui.control.namingcontainer.aspx

Events for ButtonSet

None.

Methods for ButtonSet

None.

Theming for ButtonSet

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

Sample markup with Ektron UI CSS Framework classes

```
<div class="ektron-ui-buttonSet ui-buttonset" id="buttonSet">
   <span autocomplete="off"
      class=" class=”ektron-ui-control ektron-ui-button">
      <input type="checkbox"
      name="buttonSet$uxButton1$Button$aspCheckbox"
      id="buttonSet_uxButton1_Button_aspCheckbox"
      class="ui-helper-hidden-accessible">
   </span>
   <label id="buttonSet_uxButton1_Button_aspCheckboxLabel"
```

```
      for="buttonSet_uxButton1_Button_aspCheckbox"
      aria-pressed="false"
      class="ui-button ui-widget ui-state-default
        ui-button-text-only ui-corner-left"
      role="button" aria-disabled="false">
      <span class="ui-button-text">option1</span>
   </label>

   <span autocomplete="off" class="
      class=”ektron-ui-control ektron-ui-button">
      <input type="checkbox"
      name="buttonSet$uxButton2$Button$aspCheckbox"
      id="buttonSet_uxButton2_Button_aspCheckbox"
      class="ui-helper-hidden-accessible">
   </span>
   <label id="buttonSet_uxButton2_Button_aspCheckboxLabel"
      for="buttonSet_uxButton2_Button_aspCheckbox"
      aria-pressed="false"
      class="ui-button ui-widget ui-state-default
        ui-button-text-only ui-corner-right"
      role="button"
      aria-disabled="false">
      <span class="ui-button-text">option2</span>
   </label>
</div>
```

Examples for ButtonSet

ButtonSet has the following variations:

- **Default Functionality**. Assigned a standard form input field of type button. Use standard ASP Button events to generate client-side or post back events.
- **Update Panel**. The page does a partial post back and the page fields are updated with the values added in text box.

Default functionality example

.aspx

```
<ektronUI:CssBlock ID="CssBlock1" runat="server">
    <CssTemplate>
        .mediaSet {  height:2.35em !important; }
    </CssTemplate>
</ektronUI:CssBlock>
<p>
    The media player tool bar in this example shows play, seek previous, and seek next
        buttons.
</p>
<ektronUI:ButtonSet ID="mediaSet1" runat="server">
    <ektronUI:Button CssClass="mediaSet" ID="uxRewind" PrimaryIcon="SeekPrevious"
DisplayMode="RadioButton" ToolTip="Seek Previous" runat="server" OnClick="uxRewind_
Click" />
    <ektronUI:Button CssClass="mediaSet" ID="uxPlay" PrimaryIcon="Play"
DisplayMode="RadioButton" ToolTip="Play" runat="server" OnClick="uxPlay_Click" />
```

```
    <ektronUI:Button CssClass="mediaSet" ID="uxForward" PrimaryIcon="SeekNext"
DisplayMode="RadioButton" ToolTip="Seek Next" runat="server" OnClick="uxForward_Click"
/>
</ektronUI:ButtonSet>
<asp:Label runat="server" ID="lblButtonSet"></asp:Label>
```

.aspx.cs

```
protected void uxRewind_Click(object sender, EventArgs e)
{
    lblButtonSet.Text = "<b>Rewind</b> option is selected.";
}
protected void uxPlay_Click(object sender, EventArgs e)
{
    lblButtonSet.Text = "<b>Play</b> option is selected.";
}
protected void uxForward_Click(object sender, EventArgs e)
{
    lblButtonSet.Text = "<b>Forward</b> option is selected.";
}
```

UpdatePanel example

.aspx

```
<ektronUI:CssBlock ID="uxMediaSet" runat="server">
    <CssTemplate>
        .mediaSet {  height:2.35em !important; }
    </CssTemplate>
</ektronUI:CssBlock>
<h3>Media Control Bar</h3>
<asp:ScriptManager runat="server" ID="scriptManager"></asp:ScriptManager>
<asp:UpdatePanel runat="server" id="UpdatePanel1">
    <ContentTemplate>
        <ektronUI:ButtonSet ID="mediaSet1" runat="server">
            <ektronUI:Button CssClass="mediaSet"  ID="uxRewind"
DisplayMode="RadioButton" ToolTip="Seek Previous" runat="server" OnClick="uxRewind_
Click"/>
            <ektronUI:Button CssClass="mediaSet"  ID="uxPlay" DisplayMode="RadioButton"
ToolTip="Play" runat="server" OnClick="uxPlay_Click"/>
            <ektronUI:Button CssClass="mediaSet"  ID="uxForward"
DisplayMode="RadioButton" ToolTip="Seek Next" runat="server" OnClick="uxForward_Click"/>
        </ektronUI:ButtonSet>
        <asp:Label runat="server" ID="lblButtonSet"></asp:Label>
    <ContentTemplate>
</asp:UpdatePanel>
```

.aspx.cs

```
protected void uxRewind_Click(object sender, EventArgs e)
{
    lblButtonSet.Text = "<b>Rewind</b> option is selected.";
}
protected void uxPlay_Click(object sender, EventArgs e)
{
```

```
    lblButtonSet.Text = "<b>Play</b> option is selected.";
}
protected void uxForward_Click(object sender, EventArgs e)
{
    lblButtonSet.Text = "<b>Forward</b> option is selected.";
}
```

Captcha

8.70 and higher

```
<EktronUI:Captcha>
```

Creates a Captcha control to ensure Web form submission is from a human and not another computer. The Captcha control has the following properties.

- `AlternateText`. Gets the localized Alt text.
- `BackgroundColor`. Gets or sets the BackgroundColor for the Captcha image.
- `CodeLength`. Gets or sets the CodeLength for the Captcha Image text.
- `CodeType`. Gets or sets the Captcha code type.
- `DefaultTemplate`. Gets the name of the Default Template used for rendering the TextField control.
- `ErrorMessage`. Gets or Sets the error message for incorrect captcha value.
- `FontFamily`. Gets or sets the font family for the Captcha image.
- `ForegroundColor`. Gets or sets the font color for the Captcha image.
- `HandlerPath`. Gets the path the handler used for generate Captcha image.
- `HelpToolTip`. Gets the localized tooltip for help icon.
- `HelpUrl`. Gets or sets the help URL for the help icon.
- `MismatchErrorMessage`. Gets the localized Alt text.
- `RefreshToolTip`. Gets the localized tooltip for refresh icon.
- `Selector`. Gets and Sets the selector of the control.
- `SpeakerToolTip`. Gets the localized tooltip for speech text icon.
- `Text`. Gets or sets the default text.
- `Title`. Gets the localized Title text.
- `Value`. Strongly typed value accessor.
- `Visible`. Gets or Sets the visibility of the control.

Events for Css

None.

Methods for Css

None.

Theming for Css

None.

Default functionality example

.aspx

```
<ektron:Captcha ID="captcha" runat="server" />
<p><ektronUI:Button ID="btnUI" runat="server"
   DisplayMode="Submit" Text="Submit" ></ektronUI:Button></p>
```

NOTE: .aspx.cs not required.

Css

8.50 and higher

```
<EktronUI:Css>
```

Lets you associate a Cascading Style Sheet (CSS) to an ASP.NET control.

The `<EktronUI:Css>` control offers several advantages over loading CSS statically on template pages and programatically in code.

- **CSS files are sent to the browser once**

  During a request, `<EktronUI:Css>` manages whether a CSS file was already referenced. Subsequent references to the same CSS file are ignored. During an AJAX-Postback inside an ASP UpdatePanel, `<EktronUI:Css>` manages which CSS files are already loaded by the browser. References to a CSS file during and AJAX-Postback are processed only if the CSS file is not already loaded in the browser.

- **CSS files are aggregated**

  To increase performance, CSS files referenced by `<EktronUI:Css>` controls are aggregated into a single file. Any URL references (including background-images), are automatically resolved into fully qualified paths. You may choose to "opt-out" of aggregation by setting the "Aggregate" property on the control to false. `<EktronUI:Css>` that have a Media attribute setting that is not an empty string and CSS that include at-keywords (`@import`, `@media`, and so on) are automatically dis-aggregated. You can switch off CSS aggregation for `<EktronUI:Css>` references by setting the `AllowCssAggregation` key to false in `Ektron.Cms.Framework.UI.Config`.

- **CSS files are minified**

  To increase performance, CSS files reference by `<EktronUI:Css>` controls are minified. Comments are removed except for comments marked as important by the CSS developer. If the compressor cannot minify a CSS file, it logs an error and uses the unminifed version instead.

  Example: `/*! Important Comment */`

  You can switch off CSS minification for `<EktronUI:Css>` references by setting the `AllowCssMinification` key to false in `Ektron.Cms.Framework.UI.Config`.

Events for Css

None.

Methods for Css

None.

Theming for Css

None.

Examples for Css

Css has the following variations:

- **Default Functionality**. References a CSS file that applies style rules.
- **Update Panel**. You can conditionally load CSS through AJAX by using an UpdatePanel and the Visible property. On the initial page request, set the Visible property to false. On AJAX-postback, set the Visible property to true.

NOTE: If the browser has already loaded the CSS, it will not be loaded a second time. Also, during an AJAX-postback, the CSS is sent to the browser only if the EktronUI Css control is inside an updating UpdatePanel.

Default functionality example

.aspx

```
<p>References a CSS file that applies background-color (black) and text-color (white)
rules to the paragraph below. <p>
<ektronUI:Css ID="uxCss" runat="server"
Path="~/developer/Framework/UI/Controls/EktronUI/Css/demo-ektronui-css.css" />
<asp:HyperLink ID="aspDemoCss" runat="server"
NavigateUrl="~/developer/Framework/UI/Controls/EktronUI/Css/demo-ektronui-css.css">Link
to referenced CSS file</asp:HyperLink>
```

NOTE: .aspx.cs not required.

UpdatePanel example

.aspx

```
<asp:ScriptManager ID="aspSm" runat="server" />
<p>You can conditionally load CSS through AJAX by using an UpdatePanel and the Visible
property.  On the initial page request, set the Visible property to false.  On AJAX-
postback, set the Visible property to true.</p>
<p><b>Note:</b> If the browser has already loaded the CSS, it will not be loaded a
second time. Also, during an Ajax-postback, the CSS is sent to the browser only if the
EktronUI Css control is inside an updating UpdatePanel.</p>
<asp:UpdatePanel ID="aspUpdatePanel" runat="server">
    <ContentTemplate>
        <p class="demo-ektronui-css-update-panel" style="border:1px solid
silver;padding:5px;">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque
sollicitudin lacus nec velit dictum eu dapibus odio convallis. Duis nulla risus, aliquam
in pulvinar ut, egestas vitae lectus. Nulla vitae ornare dolor. Nulla facilisi. Sed
semper ante ut mi posuere in aliquet leo adipiscing. Duis sagittis bibendum pulvinar.
Cras ornare quam a nunc facilisis dictum. Vivamus ornare erat nec orci viverra
volutpat.</p>
        <ektronUI:Css ID="uxCss" runat="server"
```

```
Path="~/developer/Framework/UI/Controls/EktronUI/Css/demo-ektronui-css.css"
Visible="false" />
        <ektronUI:Button DisplayMode="Anchor" ID="uxButton" runat="server"
OnClick="Button_Click" Text="Load Css Via Ajax In UpdatePanel"></ektronUI:Button>
    </ContentTemplate>
</asp:UpdatePanel>
<br />
<asp:HyperLink ID="aspDemoCss" runat="server"
NavigateUrl="~/developer/Framework/UI/Controls/EktronUI/Css/demo-ektronui-css.css">Link
to referenced CSS file</asp:HyperLink>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Site.Developer;

public partial class developer_Framework_UI_EktronUI_Css_UpdatePanel : SampleControlPage
{
    protected void Button_Click(object sender, EventArgs e)
    {
        uxCss.Visible = true;
    }
}
```

CssBlock

8.50 and higher

```
<EktronUI:CssBlock>
```

Lets you define and apply inline Cascading Style Sheet (CSS) rules inside any ASP.NET control without authoring invalid HTML markup. The CssBlock control renders its contents inside a `<style>` tag as a child of the `<head>` tag.

DataBinder evaluations may be used to generate dynamic CSS rules.

Events for CssBlock

None.

Methods for CssBlock

None.

Theming for CssBlock

None.

Examples for CssBlock

CssBlock has the following variations:

- **Default Functionality**. Defines inline CSS that applies rules.
- **Dynamic CSS**. The inline CSS rule is data-bound.

Default functionality example

.aspx

```
<p>Defines inline CSS that apply background-color (maroon) and text-color (white) rules
to the paragraph below via a <style> tag added to the <head> tag.</p>
<p class="demo-ektronui-css-block" style="border:1px solid silver;padding:5px;">Lorem
ipsum dolor sit amet, consectetur adipiscing elit. Quisque sollicitudin lacus nec velit
dictum eu dapibus odio convallis. Duis nulla risus, aliquam in pulvinar ut, egestas
vitae lectus. Nulla vitae ornare dolor. Nulla facilisi. Sed semper ante ut mi posuere in
aliquet leo adipiscing. Duis sagittis bibendum pulvinar. Cras ornare quam a nunc
facilisis dictum. Vivamus ornare erat nec orci viverra volutpat.</p>
<ektronUI:CssBlock ID="uxCssBlock" runat="server">
    <CssTemplate>
        p.demo-ektronui-css-block {background-color:Maroon;color:White;}
    </CssTemplate>
</ektronUI:CssBlock>
```

NOTE: .aspx.cs not required.

Dynamic CSS example

.aspx

```
<p>The inline CSS rule defining the background color of paragraph below is DataBound to
a random color generator.</p>
<p class="demo-ektronui-css-block">Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Quisque sollicitudin lacus nec velit dictum eu dapibus odio convallis. Duis nulla
risus, aliquam in pulvinar ut, egestas vitae lectus. Nulla vitae ornare dolor. Nulla
facilisi. Sed semper ante ut mi posuere in aliquet leo adipiscing. Duis sagittis
bibendum pulvinar. Cras ornare quam a nunc facilisis dictum. Vivamus ornare erat nec
orci viverra volutpat.</p>
<ektronUI:CssBlock ID="uxCssBlock" runat="server">
    <CssTemplate>
        p.demo-ektronui-css-block {background-color:<%# this.RandomColor
%>;color:#4c4c4c;border:1px solid silver;padding:5px;}
    </CssTemplate>
</ektronUI:CssBlock>
<ektronUI:Button ID="uxChangeColor" runat="server" DisplayMode="Submit" Text="Generate
Random Background Color" />
```

.aspx.cs

```
public string RandomColor
{
    get
    {
        KnownColor[] colors = (KnownColor[])Enum.GetValues(typeof(KnownColor));
        return Color.FromKnownColor(colors[GetRandomNumber(colors.Length)]).Name;
    }
}


public int GetRandomNumber(int maxValue)
{
    RandomNumberGenerator rng = RNGCryptoServiceProvider.Create();
    byte[] bytes = new byte[4];
    rng.GetBytes(bytes);
    int ranNum = BitConverter.ToInt32(bytes, 0);
    return Math.Abs(ranNum % maxValue);
}
```

DateField

8.50 and higher

```
<EktronUI:DateField>
```

Creates an input field that allows only a date value that is in a culturally sensitive format. `<EktronUI:DateField>` supports international date formats. A prompt appears in the field and disappears when you start to type in a date.

Input is validated using a built-in *DateRule* validator. Input is optional, but you can make it mandatory by adding a *RequiredRule* to the control's *ValidationRules*. The strongly typed (date) value property is the primary means to query or update the field.

Remarks

- An internal DateRule validator is applied by default.
- The value of this field is type*Date*.
- You can find the contents of this field in the Text property as a type*String*.

Events for DateField

None.

Methods for DateField

None.

Theming for DateField

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<span id="uxDateField_DateField_uxDateFieldWrapper"
  class="ektron-ui-control ektron-ui-input ektron-ui-dateField">
  <label class="inFieldLabel" for="uxDateField_DateField_aspInput"
    id="uxDateField_DateField_aspInputLabel">
      m/d/yyyy
  </label>
  <input data-ektron-validationgroup="" id="uxDateField_DateField_aspInput"
    type="text">
</span>
```

```
<label for="uxDateField_DateField_aspInput" id="uxValidationMessage_Label"
  class="ektron-ui-control ektron-ui-validationMessage ektron-ui-hidden ">
</label>
<br>
<span id="uxSubmitButton" class="ektron-ui-control ektron-ui-button">
  <button role="button" class="ui-button ui-widget ui-state-default
    ui-corner-all ui-button-text-only" id="uxSubmitButton_Button"
    title="">
  <span class="ui-button-text">Submit</span>
  </button>
</span>
```

Examples for DateField

DateField has the following variations:

- **Default Functionality**. Replace "m/d/yyyy" with a correctly formatted date (such as "12/12/2012" and blank) and click **Submit** to enter a valid date. Enter a non-valid format (such as "12/12/201" or "2012/12/12") and click **Submit** to cause the field to change color indicating that the entry is not valid.
- **Required DateField**. Replace "m/d/yyyy" with a correctly formatted date (such as "12/12/2012") and click **Submit** to enter a valid date. Enter a non-valid format (such as blank, "12/12/201" or "2012/12/12") and click **Submit** to cause the field to change color indicating that the entry is not valid.

Default functionality example

.aspx

```
<asp:Label id="uxDateFieldLabel" AssociatedControlID="uxDateField" runat="server">Ektron
Date Field:</asp:Label>
<ektronUI:DateField ID="uxDateField" ErrorMessage="Please enter a valid date."
runat="server" ></ektronUI:DateField>
<ektronUI:ValidationMessage ID="uxValidationMessage" AssociatedControlID="uxDateField"
runat="server" />
<br />
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button>
```

.aspx.cs

```
protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // if valid, then get the user supplied value
    if (Page.IsValid)
    {
        string value = uxDateField.Text;
        // at this point we could use the value with confidence...
    }
}
```

Required DateField example

.aspx

```
<asp:Label id="uxDateFieldLabel" AssociatedControlID="uxDateField" runat="server">Ektron
Date Field:</asp:Label>
<ektronUI:DateField ID="uxDateField" ErrorMessage="Please enter a valid date."
runat="server">
    <ValidationRules>
        <ektronUI:RequiredRule ErrorMessage="A date-value is required!" />
    </ValidationRules>
</ektronUI:DateField>
<ektronUI:ValidationMessage AssociatedControlID="uxDateField"
ID="uxDateFieldValidationMessage" runat="server" /><br />
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button>
```

.aspx.cs

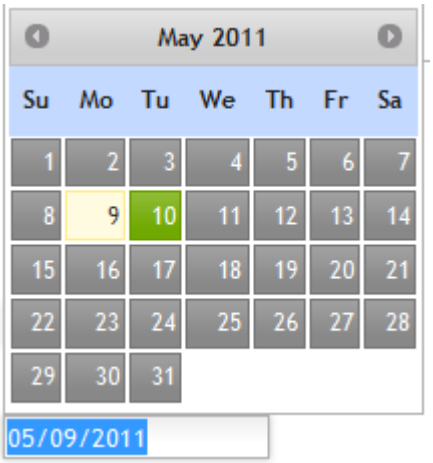
```
protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // if valid, then get the user supplied value
    if (Page.IsValid)
    {
        string value = uxDateField.Text;
        // at this point we could use the value with confidence...
    }
}
```

Datepicker

8.50 and higher

```
<EktronUI:Datepicker>
```

Lets you double click to post a date to your page.



The `<EktronUI:Datepicker>` control uses built in client-side and server-side, culturally sensitive validation to ensure that user input conforms to a valid date format.

You can customize the date format and language, restrict the selectable date ranges, and add buttons and other navigation options. By default, `<EktronUI:Datepicker>` calendar is associated with an input field and opens in a small overlay *onFocus*. It closes automatically *onBlur* or when a date is selected. It can also be displayed inline.

The `<EktronUI:Datepicker>` control provides support for localizing content for other languages and date formats. By default, `<EktronUI:Datepicker>` uses the server's default culture, but you can direct individual instances of the control for another cultural context. For more information about culture settings, see *Setting Up Locales* in the *Working with Multi-Language Content* chapter of the *Ektron Reference*.

Keyboard shortcuts

- **Ctrl/Left**. Previous day
- **Ctrl/Right**. Next day
- **Ctrl/Up**. Previous week
- **Ctrl/Down**. Next week
- **Page Up**. Previous month
- **Page Down**. Next month
- **Ctrl/Page Up**. Previous year
- **Ctrl/Page Down**. Next year
- **Ctrl/Home**. Current month or Open (when closed)

- **Ctrl/End**. Close and erase the date
- **Enter**. Accept the selected data
- **Esc**. Close the calendar without selection

Events for Datepicker

None.

Methods for Datepicker

None.

Theming for Datepicker

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div class="ektron-ui-control ektron-ui-datepicker
  ektron-ui-inline" id="uxDatepicker">
  <span class="ektron-ui-control ektron-ui-input
    ektron-ui-dateFieldkey"
    id="uxDatepicker_Datepicker_uxDateField_
      DateField_uxDateFieldWrapper">
    <label id="uxDatepicker_Datepicker_uxDateField_
      DateField_aspInputLabel"
      for="uxDatepicker_Datepicker_uxDateField_
        DateField_aspInput"
      class="inFieldLabel">
      m/d/yyyy
    </label>
    <input type="text"
      id="uxDatepicker_Datepicker_uxDateField_
        DateField_aspInput"
      name=uxDatepicker$Datepicker$uxDateField$
        DateField$aspInput" class="hasDatepicker">
  </span>
..
</div>
…
<div id="ui-datepicker-div"
  class="ui-datepicker ui-widget ui-widget-content
    ui-helper-clearfix ui-corner-all ui-helper-hidden-accessible">
  <div class="ui-datepicker-header ui-widget-header
    ui-helper-clearfix ui-corner-all">
    <a class="ui-datepicker-prev ui-corner-all"> title="Prev">
```

```
      <span class="ui-icon ui-icon-circle-triangle-w">Prev</span>
    </a>
    <a class="ui-datepicker-next ui-corner-all" title="Next">
      <span class="ui-icon ui-icon-circle-triangle-e">Next</span>
    </a>
  <div class="ui-datepicker-title">
    <span class="ui-datepicker-month">January</span>
    <span class="ui-datepicker-year">2009</span>
  </div>
</div>
<table class="ui-datepicker-calendar">
  <thead>
    <tr>
      <th class="ui-datepicker-week-end">
        <span title="Sunday">Su</span>
      </th>
      .
    </tr>
  </thead>
  <tbody>
    <tr>
       <td class="ui-datepicker-week-end ui-datepicker-other-month ">
         1
       </td>
         .
         .
         .
    </tr>
  </tbody>
</table>
<div class="ui-datepicker-buttonpane ui-widget-content">
    <button type="button"
       class="ui-datepicker-current ui-state-default
         ui-priority-secondary ui-corner-all">Today</button>
    <button type="button"
       class="ui-datepicker-close ui-state-default
         ui-priority-primary ui-corner-all">Done</button>
  </div>
</div>
```

Examples for Datepicker

DatePicker has the following variations:

- **Default Functionality**. Click on the date field to display a pop-up calendar from which you can select a date. When you click **Submit**, a confirmation message appears. You can type in a date, but it must be in a valid format. For example, enter a non-valid format (such as "12/12/201" or "2012/12/12") and the field changes color indicating that the entry is not valid.
- **Animations**. With the `ShowAnim` property it is possible to use different animations when opening or closing the Datepicker. Choose an animation from the drop-down, then click on the input to see its effect.

- **Date Range**. Click on the From and To date fields to display a pop-up calendar from which you can select a date. You can type in a date, but it must be in a valid format. For example, enter a non-valid format (such as "12/12/201" or "2012/12/12") and the field changes color indicating that the entry is not valid.
- **Display Month and Year Menus**. Show month and year drop-downs in place of the static month/year header to facilitate navigation through large timeframes. Add the boolean `changeMonth` and `changeYear` options.
- **Display Multiple Months**. Set the `numberOfMonths` option to an integer of 2 or more to show multiple months in a single Datepicker.
- **Icon Trigger**. Click the icon next to the input field to show the Datepicker. Set the Datepicker to open on focus (default behavior), on icon click, or both.
- **Inline**. Displays the Datepicker embedded in the page instead of in an overlay.
- **Localization**. The Datepicker's calendar language and format is based on the server's default culture. You can also override the default behavior for each instance of the Datepicker by specifying a culture. For example:
  1. From the Change Current Culture drop down menu, select another culture (for example, French (France) (fr-FR)).
  2. Click on the Date field and select a date from the calendar such as January 23, 2012. The date is displayed as 23/01/2012 (d/m/yyyy).

  The Datepicker also includes built-in support for languages that read right-to-left, such as Arabic and Hebrew.

  NOTE: The select element contains options for supported .NET cultures. If you select a culture that is not supported, Datepicker defaults to "English (United States) (en-US)." You can add translations by creating a new, translated culture file.

- **Restrict Date Range**. Restrict the range of selectable dates with the `minDate` and `maxDate` options. Set the beginning and end dates as actual dates (`new Date(2009, 1 - 1, 26)`), as a numeric offset from today (`-20`), or as a string of periods and units (`+1M +10D`). For the last, use `D` for days, `W` for weeks, `M` for months, or `Y` for years.
- **Show Week of the Year**. The Datepicker can show the week of the year. The default calculation follows the ISO 8601 definition: the week starts on Monday, the first week of the year contains the first Thursday of the year. This means that some days from one year may be placed into weeks 'belonging' to another year.

Default functionality example

.aspx

```
<ektronUI:Label ID="uxDatepickerLabel" AssociatedControlID="uxDatepicker" Text="Date:"
runat="server" />
<ektronUI:Datepicker ID="uxDatepicker" runat="server">
    <ValidationRules>
        <ektronUI:RequiredRule ErrorMessage="Please select a date." />
    </ValidationRules>
</ektronUI:Datepicker>
```

```
<ektronUI:ValidationMessage AssociatedControlID="uxDatepicker"
ID="uxDatepickerValidationMessage" runat="server" /><br />
<asp:Literal ID="aspLiteral" runat="server" /><br />
<ektronUI:Button DisplayMode="Button" ID="uxButton" OnClick="uxButton_click"
Text="Submit" runat="server"/>
```

.aspx.cs

```
using System;
using Ektron.Site.Developer;

public partial class Datepicker : SampleControlPage
{
    protected void uxButton_click(object sender, EventArgs e)
    {
        // get the textfield value
        aspLiteral.Text = "You selected " + uxDatepicker.Value.ToLongDateString();
    }
}
```

Animations example

.aspx

```
<p>With the <code>ShowAnim</code> it is possible to use different animations when
opening or closing the datepicker.  Choose an animation from the dropdown, then click on
the input to see its effect.</p>
<ektronUI:Label ID="uxDatepickerLabel" AssociatedControlID="uxDatepicker" Text="Date:"
runat="server" />
<ektronUI:Datepicker ID="uxDatepicker" runat="server" />
<ektronUI:Label ID="aspAnimationsLabel" AssociatedControlID="aspAnimations"
runat="server" />
<asp:DropDownList ID="aspAnimations" AutoPostBack="true"
OnSelectedIndexChanged="aspAnimations_change"  runat="server">
    <asp:ListItem Value="show" Text="Show (default)" />
    <asp:ListItem Value="slideDown" Text="Slide Down"/>
    <asp:ListItem Value="fadeIn" Text="Fade in" />
    <asp:ListItem Value="blind" Text="Blind" />
    <asp:ListItem Value="clip" Text="Clip" />
    <asp:ListItem Value="drop" Text="Drop" />
    <asp:ListItem Value="fold" Text="Fold" />
    <asp:ListItem Value="highlight" Text="Highlight" />
    <asp:ListItem Value="pulsate" Text="Pulsate" />
    <asp:ListItem Value="scale" Text="Scale" />
    <asp:ListItem Value="slide" Text="Slide" />
    <asp:ListItem Value="" Text="None" />
</asp:DropDownList>
```

.aspx.cs

```
using System;
using Ektron.Site.Developer;

public partial class Datepicker_RestrictDateRange : SampleControlPage
{
```

```
    protected void aspAnimations_change(Object sender, EventArgs e)
    {
        if (String.IsNullOrEmpty(aspAnimations.Text))
        {
            uxDatepicker.ShowAnim =
Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets.DatepickerBase.ShowAnimOptions.none;
        }
        else
        {
            uxDatepicker.ShowAnim =
(Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets.DatepickerBase.ShowAnimOptions)Enum.P
arse(typeof
(Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets.DatepickerBase.ShowAnimOptions),
aspAnimations.Text, true);
        }
    }
}
```

Date Range example

.aspx

```
<p>Date ranges are easy to accomplish using two datepicker controls in tandem.</p>
<ektronUI:Label ID="uxFrom" AssociatedControlID="uxDatepickerFrom" Text="From"
runat="server" />
<ektronUI:Datepicker ID="uxDatepickerFrom" runat="server" />
<ektronUI:Label ID="uxTo" AssociatedControlID="uxDatepickerTo" Text="to" runat="server"
/>
<ektronUI:Datepicker ID="uxDatepickerTo" runat="server" />
```

NOTE: .aspx.cs not required.

Display Month and Year example

.aspx

```
<p>Show month and year dropdowns in place of the static month/year header to facilitate
navigation through large timeframes.  Add the boolean <code>changeMonth</code> and
<code>changeYear</code> options.</p>
<ektronUI:Label ID="uxDatepickerLabel" AssociatedControlID="uxDatepicker" Text="Date:"
runat="server" />
<ektronUI:Datepicker ID="uxDatepicker" DisplayChangeMonth="true"
DisplayChangeYear="true" runat="server" />
```

NOTE: .aspx.cs not required.

Display Multiple Months example

.aspx

```
<p>Set the <code>numberOfMonths</code> option to an integer of 2 or more to show
multiple months in a single datepicker.</p>
<ektronUI:Label ID="uxDatepickerLabel" AssociatedControlID="uxDatepicker" Text="Date:"
```

```
runat="server" />
<ektronUI:Datepicker ID="uxDatepicker" NumberOfMonths="3" runat="server" />
```

NOTE: .aspx.cs not required.

Icon Trigger example

.aspx

```
<p>Click the icon next to the input field to show the datepicker.  Set the datepicker to
open on focus (default behavior), on icon click, or both.</p>
<ektronUI:Label ID="uxDatepickerLabel" AssociatedControlID="uxDatepicker" Text="Date:"
runat="server" />
<ektronUI:Datepicker ID="uxDatepicker" ButtonImageOnly="true" ShowOn="Button"
runat="server" />

<ektronUI:CssBlock ID="uxCssBlock" runat="server">
    <CssTemplate>
        img.ui-datepicker-trigger {vertical-align: middle; cursor: pointer;}
    </CssTemplate>
</ektronUI:CssBlock>
```

.aspx.cs

```
using System;
using Ektron.Cms.Framework.UI;
using Ektron.Cms.Interfaces.Context;
using Ektron.Site.Developer;

public partial class Datepicker_IconTrigger : SampleControlPage
{
    protected void Page_Init(object sender, EventArgs e)
    {
        ICmsContextService cmsContextService = ServiceFactory.CreateCmsContextService();
        uxDatepicker.ButtonImage = cmsContextService.WorkareaPath +
"/images/ui/icons/calendar.png";
    }
}
```

Inline example

.aspx

```
<ektronUI:Label ID="uxDatepickerLabel" AssociatedControlID="uxDatepicker" Text="Date:"
runat="server" />
<ektronUI:Datepicker ID="uxDatepicker" DisplayMode="Inline" runat="server" />
```

NOTE: .aspx.cs not required.

Localization example

.aspx

```
<asp:Label ID="aspChangeCultureLabel" AssociatedControlID="aspChangeCulture"
runat="server">Change Current Culture:</asp:Label>
<asp:DropDownList id="aspChangeCulture" name="culture"  AutoPostBack="true"
OnSelectedIndexChanged="aspChangeCulture_change" runat="server"></asp:DropDownList>
<br />
<br />
<ektronUI:Label ID="uxDatepickerLabel" AssociatedControlID="uxDatepicker" Text="Date:"
runat="server" />
<ektronUI:Datepicker ID="uxDatepicker" DisplayMode="Default" runat="server" />
```

.aspx.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Globalization;
using System.Threading;
using System.Web.UI.WebControls;
using Ektron.Site.Developer;

public partial class Datepicker_Localization : SampleControlPage
{
    protected override void OnPreRender(EventArgs e)
    {
        SortDropDown(aspChangeCulture);
        base.OnPreRender(e);
    }

    protected void Page_Init(object sender, EventArgs e)
    {
        Array cultureTypes = CultureInfo.GetCultures(CultureTypes.AllCultures);

        if (!IsPostBack)
        {
            foreach (CultureInfo cultureInfo in cultureTypes)
            {
                ListItem item = new ListItem();
                item.Text = cultureInfo.EnglishName + " (" + cultureInfo.Name + ")";
                item.Value = cultureInfo.Name;
                if (cultureInfo.Name == "en-US")
                {
                    item.Selected = true;
                }
                else
                {
                    item.Selected = false;
                }
                aspChangeCulture.Items.Add(item);
                item = null;
            }
        }
```

```
    }

    protected void aspChangeCulture_change(Object sender, EventArgs e)
    {
        Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture
(aspChangeCulture.SelectedValue);
    }

    #region helpers
    public void SortDropDown(DropDownList cbo)
    {
        List<ListItem> lstListItems = new List<ListItem>();
        foreach (ListItem li in cbo.Items)
        {
            lstListItems.Add(li);
        }
        lstListItems.Sort(new ListItemComparer());
        cbo.Items.Clear();
        cbo.Items.AddRange(lstListItems.ToArray());
    }


    public class ListItemComparer : IComparer<ListItem>
    {

        public int Compare(ListItem x, ListItem y)
        {
            CaseInsensitiveComparer c = new CaseInsensitiveComparer();
            return c.Compare(x.Text, y.Text);
        }
    }
    #endregion
}
```

Restrict Date Range example

.aspx

```
<ektronUI:Label ID="uxDatepickerLabel" AssociatedControlID="uxDatepicker" Text="Date:"
runat="server" />
<ektronUI:Datepicker ID="uxDatepicker" MinDate="01/01/2011" MaxDate="09/10/2011"
runat="server" />
```

NOTE: .aspx.cs not required.

Show Week of Year example

.aspx

```
<ektronUI:Label ID="uxDatepickerLabel" AssociatedControlID="uxDatepicker" Text="Date:"
runat="server" />
<ektronUI:Datepicker ID="uxDatepicker" ShowWeek="true" runat="server" />
```

NOTE: .aspx.cs not required.

DecimalField

8.50 and higher

```
<EktronUI:DecimalField>
```

Creates an input field that allows only decimal values.

Numbers are rounded to the nearest specified decimal place. For example, if you specify 2 decimal places (the default) and type 0.337, the number rounds to 0.34.

User-supplied input is validated using the field's built-in *DecimalRule* validator. Input is optional, but you can make it mandatory by adding a *RequiredRule* to the control's *ValidationRules*. The strongly typed (decimal) value property is the primary means to query or update the field.

Events for DecimalField

None.

Methods for DecimalField

None.

Theming for DecimalField

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<span class="ektron-ui-control ektron-ui-input
  ektron-ui-decimalField"
  id="uxDecimalField_DecimalField_uxTextField">
  <span class="ui-spinner ui-state-default ui-widget
    ui-widget-content ui-corner-all">
    <input type="text" autocomplete="off"
      id="uxDecimalField_DecimalField_aspInput"
      value="0.0"
      name="uxDecimalField$DecimalField$aspInput"
      class="ui-spinner-input" role="spinbutton"
      aria-valuemin="-7.922816251426434e+28"
      aria-valuemax="7.922816251426434e+28"
      aria-valuenow="0"
```

```
      data-ektron-initialvalue="0.0"
      data-ektron-validationgroup="" />
    <a class="ui-spinner-button ui-spinner-up
      ui-corner-tr ui-button ui-widget
      ui-state-default ui-button-text-only"
      tabindex="-1" role="button">
      <span class="ui-button-text">
        <span class="ui-icon
          ui-icon-triangle-1-n">?
        </span>
      </span>
    </a>
    <a class="ui-spinner-button ui-spinner-down
      ui-corner-br ui-button ui-widget
      ui-state-default ui-button-text-only"
      tabindex="-1" role="button">
      <span class="ui-button-text">
        <span class="ui-icon
          ui-icon-triangle-1-s">?
        </span>
      </span>
    </a>
  </span>
</span>
```

Examples for DecimalField

DecimalField has the following variations:

- **Default Functionality**. Lets you enter decimal values with a specified number of decimal places. If you enter a decimal value with more decimal places than allowed and click **Submit**, the number is rounded to the nearest specified decimal place. For example, 0.337 is rounded to 0.34 when 2 decimal places are specified.
- **Min-Max DecimalField**. Lets you enter only decimal values within a specified minimum and maximum range. If you enter a decimal value outside the range, the number automatically resets to the nearest limit.
- **Precision DecimalField**. Sets DecimalPlaces to a specific number. The default value is 2. For example, if you specify 6 and enter fewer than 6 places, zeroes are added; 0.123 is replaced with 0.123000 when you click **Submit**. If you enter more decimal values such as 0.12345678 and click **Submit**, the number is rounded to the nearest 6 decimal places (0.123457).

Default functionality example

.aspx

```
<asp:Label id="uxDecimalFieldLabel" AssociatedControlID="uxDecimalField"
runat="server">Ektron Decimal Field:</asp:Label>
<ektronUI:DecimalField ID="uxDecimalField" ErrorMessage="Please enter a valid decimal."
runat="server" ></ektronUI:DecimalField>
<ektronUI:ValidationMessage ID="uxValidationmessage"
```

```
AssociatedControlID="uxDecimalField" runat="server" />
<br />
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button>
```

.aspx.cs

```
protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // if valid, then get the user supplied value
    if (Page.IsValid)
    {
        string value = uxDecimalField.Text;
        // at this point we could use the value with confidence...
    }
}
```

Min-Max DecimalField example

.aspx

```
<asp:Label id="uxDecimalFieldLabel" AssociatedControlID="uxDecimalField"
runat="server">Enter a value from 0 to 1.00:</asp:Label>
<ektronUI:DecimalField ID="uxDecimalField" ErrorMessage="Please enter a valid decimal."
MinValue="0" MaxValue="1.00" runat="server" ></ektronUI:DecimalField>
<ektronUI:ValidationMessage ID="uxValidationmessage" AssociatedControlID=""
runat="server" /><br />
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button>
```

.aspx.cs

```
protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // if valid, then get the user supplied value
    if (Page.IsValid)
    {
        string value = uxDecimalField.Text;
        // at this point we could use the value with confidence...
    }
}
```

Precision DecimalField example

.aspx

```
<asp:Label id="uxDecimalFieldLabel" AssociatedControlID="uxDecimalField"
runat="server">Ektron Decimal Field:</asp:Label>
<ektronUI:DecimalField ID="uxDecimalField" ErrorMessage="Please enter a valid decimal."
DecimalPlaces="6" runat="server" ></ektronUI:DecimalField>
<ektronUI:ValidationMessage ID="uxValidationmessage"
AssociatedControlID="uxDecimalField" runat="server" />
<br />
```

```
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button>
```

.aspx.cs

```
protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // if valid, then get the user supplied value
    if (Page.IsValid)
    {
        string value = uxDecimalField.Text;
        // at this point we could use the value with confidence...
    }
}
```

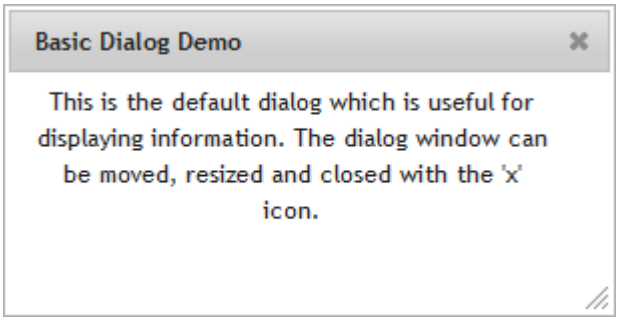
Dialog

8.50 and higher

```
<EktronUI:Dialog>
```

Creates a floating window that contains a title bar, content area, and dialog buttons. A dialog window can be moved, resized, and closed with the "x" icon. If the content length exceeds maximum height, a scroll bar automatically appears.

The following image shows an `<EktronUI:Dialog>` box that contains a `ContentBlock` server control. Additionally, this dialog box is set to *modal* with *vertical scrolling*, *resizable*, and *draggable* options turned on.



Events for Dialog

None.

Methods for Dialog

None.

Theming for Dialog

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div style="display: block; z-index: 1002;
  outline: 0px none; position: absolute;
  height: auto; width: 300px; top: 108px;
  left: 352px;" class="ui-dialog ui-widget
  ui-widget-content ui-corner-all ui-draggable
  ui-resizable" tabindex="-1" role="dialog"
```

```
aria-labelledby="ui-dialog-title-uxDialog_
  Dialog_uxDialogControl">
<div class="ui-dialog-titlebar ui-widget-header
  ui-corner-all ui-helper-clearfix">
  <span class="ui-dialog-title"
    id="ui-dialog-title-uxDialog_Dialog_
      uxDialogControl">PostBack Dialog Demo
  </span>
  <a href="#" class="ui-dialog-titlebar-close
    ui-corner-all" role="button">
    <span class="ui-icon ui-icon-closethick">
      close
     </span>
  </a>
</div>
<div class="ektron-ui-dialog ui-dialog-content
  ui-widget-content"
  id="uxDialog_Dialog_uxDialogControl"
  style="width: auto; min-height: 48.8px;
  height: auto;">
    Dialog Content Goes Here.
</div>
<div class="ui-resizable-handle ui-resizable-n">
</div>
<div class="ui-resizable-handle ui-resizable-e">
</div>
<div class="ui-resizable-handle ui-resizable-s">
</div>
<div class="ui-resizable-handle ui-resizable-w">
</div>
<div class="ui-resizable-handle
  ui-resizable-se ui-icon
  ui-icon-gripsmall-diagonal-se
  ui-icon-grip-diagonal-se"
  style="z-index: 1001;">
</div>
<div class="ui-resizable-handle
  ui-resizable-sw" style="z-index: 1002;">
</div>
<div class="ui-resizable-handle ui-resizable-ne"
  style="z-index: 1003;">
</div>
<div class="ui-resizable-handle ui-resizable-nw"
  style="z-index: 1004;">
</div>
<div class="ui-dialog-buttonpane ui-widget-content
  ui-helper-clearfix">
  <div class="ui-dialog-buttonset">
    <button type="button" class="ui-button
      ui-widget ui-state-default ui-corner-all
      ui-button-text-only" role="button" >
      <span class="ui-button-text">
        Dialog Button Text
      </span>
    </button>
  </div>
```

```
  </div>
</div>
```

Examples for Dialog

Dialog has the following variations:

- **Default Functionality**. Displays a floating window. The dialog window can be moved, resized and closed with the X icon.
- **Client Controlled**. Displays a floating window when you click a button.
- **Modal Dialog**. The dialog is a child window that requires users to interact with it before they can perform any action on the parent page. It adds an overlay that makes the dialog look more prominent because it dims out the page content.
- **Non-Draggable Dialog**. A basic dialog that you cannot drag; `Draggable="false"`.
- **Non-Resizable Dialog**. A basic dialog that you cannot resize; `Resizable="false"`.
- **Postback Dialog**. A dialog that has a postback event when you click a button.
- **Stackable Dialog**. Stackable dialogs move to the front of the other dialogs when it has focus. Click on the others or move dialogs around. Non-Stackable dialogs will not move to the front of stackable dialogs.

Default functionality example

.aspx

```
<ektronUI:Dialog ID="uxDialog" runat="server" Title="Basic Dialog Demo" AutoOpen="true"
>
    <ContentTemplate>
        This is the default dialog which is useful for displaying information. The
dialog window can be moved, resized and closed with the 'x' icon.
    </ContentTemplate>
</ektronUI:Dialog>
```

NOTE: .aspx.cs not required.

Client Controlled example

.aspx

```
<ektronUI:JavaScriptBlock ID="uxOpenDialogScript" runat="server"
ExecutionMode="OnParse">
    <ScriptTemplate>
        function openDialog(){
            $ektron("<%= uxDialog.Selector %>").dialog("open");
        }

        function closeDialog(){
            $ektron("<%= uxDialog.Selector %>").dialog("close");
        }
```

```
    </ScriptTemplate>
</ektronUI:JavaScriptBlock>
<p>Click the "Open Dialog" button to start the demo.</p>
<ektronUI:Button DisplayMode="Button" runat="server" ID="OpenDialogButton" Text="Open
Dialog" OnClientClick="openDialog(); return false;"></ektronUI:Button>
<br />
<ektronUI:Button DisplayMode="Button" runat="server" ID="CloseDialogButton" Text="Close
Dialog" OnClientClick="closeDialog();return false;"></ektronUI:Button>
<ektronUI:Dialog ID="uxDialog" runat="server" Title="Client Controlled Dialog Demo"
AutoOpen="false" >
    <ContentTemplate>
        A basic dialog that is opened and closed from client-side Javascript code,
called by each buttons client-side onclick handler.
    </ContentTemplate>
    <Buttons>
        <ektronUI:DialogButton ID="DialogButtonClose" runat="server" Text="Close Dialog"
CloseDialog="true"></ektronUI:DialogButton>
    </Buttons>
</ektronUI:Dialog>
```

NOTE: .aspx.cs not required.

Modal Dialog example

.aspx

```
<ektronUI:Dialog ID="uxDialog" runat="server" Title="Modal Dialog Demo" AutoOpen="true"
Modal="true" >
    <ContentTemplate>
        <p>In modal mode, the dialog is a child window that requires users to interact
with it before they can perform any action on the parent page.</p>
        <p>It adds an overlay that makes the dialog look more prominent because it dims
out the page content.</p>
    </ContentTemplate>
</ektronUI:Dialog>
```

NOTE: .aspx.cs not required.

Non-Draggable Dialog example

.aspx

```
<ektronUI:Dialog ID="uxDialog" runat="server" Title="Non-Draggable Dialog"
AutoOpen="true" Draggable="false" >
    <ContentTemplate>
        A basic dialog that is not draggable; Draggable="false".
    </ContentTemplate>
</ektronUI:Dialog>
```

NOTE: .aspx.cs not required.

Non-Resizable Dialog example

.aspx

```
<ektronUI:Dialog ID="uxDialog" runat="server" Title="Non-Resizable Dialog"
AutoOpen="true" Resizable="false" >
    <ContentTemplate>
        A basic dialog that is not resizable; Resizable="false".
    </ContentTemplate>
</ektronUI:Dialog>
```

NOTE: .aspx.cs not required.

Postback example

.aspx

```
<ektronUI:Dialog ID="uxDialog" runat="server" Title="PostBack Dialog Demo"
AutoOpen="<%#AutoOpen%>" >
    <ContentTemplate>
        Demonstrates using a server-side event handler for the DialogButton's click
event.
    </ContentTemplate>
    <Buttons>
        <ektronUI:DialogButton ID="PostBackDialogButton" runat="server" Text="Post Back"
OnClick="PostBackDialogButton_click" CloseDialog="true" />
    </Buttons>
</ektronUI:Dialog>
<asp:Literal ID="literal1" runat="server" />
```

.aspx.cs

```
protected void PostBackDialogButton_click(object sender, EventArgs e)
{
    literal1.Text = "The Dialogs Post-Back button was clicked!";
    AutoOpen = false;
}

protected override void OnPreRender(EventArgs e)
{
    DataBind();
    base.OnPreRender(e);
}
```

Stackable Dialog example

.aspx

```
<ektronUI:Dialog ID="Dialog1" runat="server" Title="Stackable Dialog One"
AutoOpen="true" Stack="true" Width="500" Height="150" >
    <ContentTemplate>
        Stackable Dialog One - Stackable dialogs move to the front of the other dialogs
when it has focus. Click on the others or move dialogs around to demonstrate this.
    </ContentTemplate>
```

```
</ektronUI:Dialog>
<ektronUI:Dialog ID="Dialog2" runat="server" Title="Stackable Dialog Two"
AutoOpen="true" Width="300" Height="200" >
    <ContentTemplate>
        Stackable Dialog Two (stackable by default) - Stackable dialogs move to the
front of the other dialogs when it has focus. Click on the others or move dialogs around
to demonstrate this.
    </ContentTemplate>
</ektronUI:Dialog>
<ektronUI:Dialog ID="Dialog" runat="server" Title="Non-Stackable Dialog" AutoOpen="true"
Stack="false" Width="700" Height="250" >
    <ContentTemplate>
        Non-Stackable Dialog - Non-Stackable dialogs will not move to the front of
stackable dialogs.
    </ContentTemplate>
</ektronUI:Dialog>
```

NOTE: .aspx.cs not required.

InfoTip

9.00 and higher

```
<EktronUI:InfoTip>
```

InfoTip initially hides a message and makes it visible "on demand." Displaying the message is triggered by hovering over the "I" image anchor. When the mouse is moved away from the icon, the message disappears.

InfoTip should describe only the UI element next to which it is placed.

Use InfoTip if help text is too long for display below an input field or if the information provided is relevant to, but not directly required for the user to successfully complete the form field.

Events for InfoTip

None.

Methods for InfoTip

None.

Theming for InfoTip

Infotip is a specific use of the jQuery UI Tooltip, and so it follows their CSS Framework and the ThemeRoller tool to create and download custom themes that are easy to build and maintain.

Example for InfoTip

Default functionality example

.aspx

```
<ektronUI:Infotip ID="uxDemo" runat="server">
    This is cryptic, because it serves no independent function.  It is present
    merely as a demonstration of the Infotip server control.  Marking it as checked
    or unchecked will have no effect on the page.
</ektronUI:Infotip>
```

NOTE: .aspx.cs not required.

Positioning example

.aspx

```
<ektronUI:Infotip ID="Infotip1" PositionMy="right bottom" PositionAt="left top"
runat="server">
```

```
    The <strong>right bottom</strong> corner of this message is positioned at the
    <strong>left top</strong> corner of the image anchor.
</ektronUI:Infotip>
```

NOTE: .aspx.cs not required.

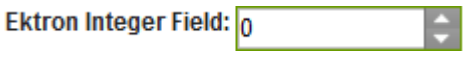
IntegerField

8.50 and higher

```
<EktronUI:IntegerField>
```

Creates an input field that allows only integers.

If you type a decimal value and press **Enter**, the number is rounded to the nearest integer. For example, if you type 12.888 and click **Submit**, the value changes to 13.



User-supplied input is validated using the field's built-in *IntegerRule* validator. Input is optional, but you can make it mandatory by adding a *RequiredRule* to the control's *ValidationRules*. The strongly typed (integer) value property is the primary means to query or update the field.

Events for IntegerField

None.

Methods for IntegerField

None.

Theming for IntegerField

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<span class="ektron-ui-control ektron-ui-input
  ektron-ui-integerField"
  id="uxIntegerField_IntegerField_uxTextField">
  <span class="ui-spinner ui-state-default
    ui-widget ui-widget-content ui-corner-all">
    <input type="text"
      id="uxIntegerField_IntegerField_aspInput"
      value="0"
      name="uxIntegerField$IntegerField$aspInput"
      class="ui-spinner-input   ektron-ui-valid"
      autocomplete="off" role="spinbutton"
      aria-valuemin="-2147483648"
      aria-valuemax="2147483647"
```

```
      aria-valuenow="0"
      data-ektron-initialvalue="0"
      data-ektron-validationgroup="" />
    <a class="ui-spinner-button ui-spinner-up
      ui-corner-tr ui-button ui-widget
      ui-state-default ui-button-text-only"
      tabindex="-1" role="button">
      <span class="ui-button-text">
        <span class="ui-icon ui-icon-triangle-1-n">?
        </span>
      </span>
    </a>
    <a class="ui-spinner-button ui-spinner-down
       ui-corner-br ui-button ui-widget
      ui-state-default ui-button-text-only"
      tabindex="-1" role="button">
      <span class="ui-button-text">
        <span class="ui-icon
          ui-icon-triangle-1-s">?
        </span>
      </span>
    </a>
  </span>
</span>
```

Examples for IntegerField

IntegerField has the following variations:

- **Default Functionality**. IntegerField lets you enter only integer values. If you enter a decimal value and click **Submit**, the number is rounded to the nearest integer.
- **Min-Max DecimalField**. Lets you enter only integer values within a specified minimum and maximum range. If you enter a decimal value and click **Submit**, the number is rounded to the nearest integer. If you enter a number outside the range, the number automatically resets to the nearest limit.

Default functionality example

.aspx

```
<asp:Label id="uxIntegerFieldLabel" AssociatedControlID="uxIntegerField"
runat="server">Ektron Integer Field:</asp:Label>
<ektronUI:IntegerField ID="uxIntegerField" ErrorMessage="Please enter a valid integer."
runat="server" ></ektronUI:IntegerField>
<ektronUI:ValidationMessage ID="uxValidationmessage"
AssociatedControlID="uxIntegerField" runat="server" />
<br />
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button>
```

.aspx.cs

```
protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // if valid, then get the user supplied value
    if (Page.IsValid)
    {
        string value = uxIntegerField.Text;
        // at this point we could use the value with confidence...
    }
}
```

Min-Max IntegerField example

.aspx

```
<asp:Label id="uxIntegerFieldLabel" AssociatedControlID="uxIntegerField"
runat="server">Enter a value from 1 to 10:</asp:Label>
<ektronUI:IntegerField ID="uxIntegerField" MinValue="1" MaxValue="10" Text="5"
ErrorMessage="Please enter a valid integer." runat="server" ></ektronUI:IntegerField>
<ektronUI:ValidationMessage ID="uxValidationmessage"
AssociatedControlID="uxIntegerField" runat="server" />
<br />
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button>
```

.aspx.cs

```
protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // if valid, then get the user supplied value
    if (Page.IsValid)
    {
        string value = uxIntegerField.Text;
        // at this point we could use the value with confidence...
    }
}
```

JavaScript

8.50 and higher

```
<EktronUI:JavaScript>
```

Associates an external file containing JavaScript (JS) code to an ASP.NET control. The `<EktronUI:JavaScript>` control offers several advantages over loading JavaScript statically on template pages and programmatically in code

- **JS files are sent to the browser once**. During a request, `<EktronUI:JavaScript>` manages whether a JS file was already referenced. Subsequent references to the same JS file are ignored. During an AJAX-Postback inside an ASP UpdatePanel, `<EktronUI:JavaScript>` manages which JS files are already loaded by the browser. References to a JS file during and AJAX-Postback are processed only if the JS is not already loaded in the browser.
- **JS files are aggregated**. To increase performance, JS files referenced by `<EktronUI:JavaScript>` controls are aggregated into a single file. Any URL references (including background-images), are automatically resolved into fully qualified paths. You may choose to "opt-out" of aggregation by setting the `Aggregate` property on the control to `false`. You can switch off JS aggregation for `<EktronUI:JavaScript>` references by setting the `AllowJavaScriptAggregation` key to `false` in `Ektron.Cms.Framework.UI.Config`.
- **JS files are minified**. To increase performance, JS files reference by `<EktronUI:JavaScript>` controls are minified. You can switch off JS minification for `<EktronUI:JavaScript>` references by setting the `AllowJavaScriptMinification` key to `false` in `Ektron.Cms.Framework.UI.Config`.

Events for JavaScript

None.

Methods for JavaScript

None.

Theming for JavaScript

None.

Example for JavaScript

Default functionality example

.aspx

```
<div class="targetForDemoContent"> </div>
<ektronUI:Button ID="uxAddButton" CssClass="demoAddButton" Text="Click Me"
DisplayMode="Button" ToolTip="Click Me" runat="server" />
<ektronUI:Button ID="uxResetButton" CssClass="demoResetButton" Text="Reset Demo"
DisplayMode="Button" ToolTip="Reset Demo" runat="server" />
<ektronUI:JavaScript ID="uxJavaScript" runat="server" Path="
{SitePath}/developer/Framework/UI/Controls/EktronUI/JavaScript/remoteJavaScriptSample.j
s" />
<ektronUI:CssBlock ID="uxCssBlock" runat="server">
    <CssTemplate>
        .targetForDemoContent {padding: .5em; margin: .5em 0; border: solid #ccc 1px;
background: #eee; color: #666666;}
    </CssTemplate>
</ektronUI:CssBlock>
```

NOTE: .aspx.cs not required.

JavaScriptBlock

8.50 and higher

```
Tag: <EktronUI:JavaScriptBlock>
```

Lets you define and apply inline JavaScript (JS) rules inside any ASP.NET control without authoring invalid HTML markup. The JavaScriptBlock control renders its contents inside a `<script>` tag and can have the code within the block execute either as it is parsed by the browser, or after the rendered page is ready.

DataBinder evaluations may be used to generate dynamic JavaScript code.

Events for JavaScriptBlock

None.

Methods for JavaScriptBlock

None.

Theming for JavaScriptBlock

None.

Example for JavaScriptBlock

Default functionality example

.aspx

```
<div class="targetForDemoContent"> </div>
<ektronUI:Button ID="uxAddButton" CssClass="demoAddButton" Text="Click Me"
DisplayMode="Button" ToolTip="Click Me" runat="server" />
<ektronUI:Button ID="uxResetButton" CssClass="demoResetButton" Text="Reset Demo"
DisplayMode="Button" ToolTip="Reset Demo" runat="server" />
<ektronUI:CssBlock ID="uxCssBlock" runat="server">
    <CssTemplate>
        .targetForDemoContent {padding: .5em; margin: .5em 0; border: solid #ccc 1px;
background: #eee; color: #666666;}
    </CssTemplate>
</ektronUI:CssBlock>
<ektronUI:JavaScriptBlock ID="uxJavaScriptBlock" runat="server">
    <ScriptTemplate>
        var targetForDemoContent = $ektron(".targetForDemoContent");
        $ektron(".demoAddButton").bind("click", function () {
            targetForDemoContent.html("Thanks for clicking that button and adding this
sentence to the page.");
            return false;
        });
```

```
        $ektron(".demoResetButton").bind("click", function () {
            targetForDemoContent.html(" ");
            return false;
        });
    </ScriptTemplate>
</ektronUI:JavaScriptBlock>
```

NOTE: .aspx.cs not required.

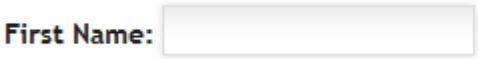
Label

8.50 and higher

```
Tag: <EktronUI:Label>
```

Outputs an HTML <label> tag that is different than the tag in a standard ASP label control. <EktronUI:label> sets the "for" attribute of the <label> tag property (differently from the standard ASP .NET <label> control).

The following image shows "First Name" label next to a text field.



Use this control where you need to associate this label with another EktronUI server control instead of an asp:label because an asp:label will not set the for attribute correctly when the AssociatedControlID is set on an EktronUI control. The markup is always an HTML <label> tag. An asp:Label outputs either a or a <label> tag depending on whether the AssociatedControlID property is set or not.

Validation Rules you can apply to this control are:

- CustomRule
- RegexRule
- RequiredRule

Properties

- CssClass. Gets or sets the Cascading Style Sheet (CSS) class rendered by the control on the client.
- Enabled. Gets or sets a value indicating whether the control is enabled.
- ID. Gets or sets the programmatic identifier assigned to the server control.
- TabIndex. Gets or sets the tab index of the control.
- TextToolTip. Gets or sets the text displayed when the mouse pointer hovers over the control.
- ViewStateMode. Gets or sets the view-state mode of this control.
- Visible. Gets or sets a value that indicates whether a control is rendered as UI on the page.

Events for Label

None.

Methods for Label

None.

Theming for Label

None.

Example for Label

Default functionality example

.aspx

```
<ektronUI:Label runat="server" ID="uxLabel" AssociatedControlID="uxTextField">Ektron UI
Label:</ektronUI:Label>
<ektronUI:TextField runat="server" ID="uxTextField"></ektronUI:TextField>
```

NOTE: .aspx.cs not required.

Message

8.50 and higher

```
<EktronUI:Message>
```

Displays messages to users of an application in a consistent and themable format. The control supports 5 built-in display modes for messages: Error, Help, Information, Success and Warning. By default the control displays a message as type Error. The control provides easily accessible properties to let you quickly change both the content and display type for a message, allowing a single instance of the control to serve multiple UI purposes.

Events for Message

None.

Methods for Message

None.

Theming for Message

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div class="ektron-ui-control ui-corner-all ektron-ui-message ektron-ui-error ektron-ui-
clearfix"
  id="uxErrorMessage_Message_uxMessage">
  <span class="ui-icon ektron-ui-sprite ui-icon-notice ektron-ui-sprite-exclamation"
    id="uxErrorMessage_Message_aspIcon">
  </span>
  <div class="ektron-ui-clearfix
    ektron-ui-messageBody">
    An example of an <code>Error</code> message.
  </div>
</div>
<div class="ektron-ui-control ui-corner-all ektron-ui-message ektron-ui-help ektron-ui-
clearfix"
  id="uxHelpMessage_Message_uxMessage">
  <span class="ui-icon ektron-ui-sprite ui-icon-help ektron-ui-sprite-help"
    id="uxHelpMessage_Message_aspIcon">
  </span>
```

```
  <div class="ektron-ui-clearfix
    ektron-ui-messageBody">
    An example of a <code>Help</code> message.
  </div>
 </div>
 <div class="ektron-ui-control ui-corner-all ektron-ui-message ektron-ui-information
ektron-ui-clearfix"
  id="uxInformationMessage_Message_uxMessage">
  <span class="ui-icon ektron-ui-sprite ui-icon-info ektron-ui-sprite-information"
    id="uxInformationMessage_Message_aspIcon">
  </span>
  <div class="ektron-ui-clearfix
    ektron-ui-messageBody">
    An example of an <code>Information</code>
    message.
  </div>
</div>
<div class="ektron-ui-control ui-corner-all ektron-ui-message ektron-ui-success ektron-
ui-clearfix"
  id="uxSuccessMessage_Message_uxMessage">
  <span class="ui-icon ektron-ui-sprite ui-icon-circle-check ektron-ui-sprite-accept"
    id="uxSuccessMessage_Message_aspIcon">
  </span>
  <div class="ektron-ui-clearfix
    ektron-ui-messageBody">
    An example of a <code>Success</code> message.
  </div>
</div>
<div class="ektron-ui-control ui-corner-all ektron-ui-message ui-state-highlight
  ektron-ui-warning ektron-ui-clearfix"
  id="uxWarningMessage_Message_uxMessage">
  <span class="ui-icon ektron-ui-sprite ui-icon-alert ektron-ui-sprite-warning"
    id="uxWarningMessage_Message_aspIcon">
  </span>
  <div class="ektron-ui-clearfix
    ektron-ui-messageBody">
    An example of a <code>Warning</code> message.
  </div>
</div>
```

Sample Message types

Error

Error

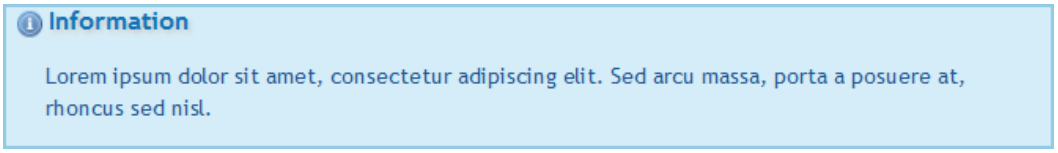
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed arcu massa, porta a posuere at, rhoncus sed nisl.

Help

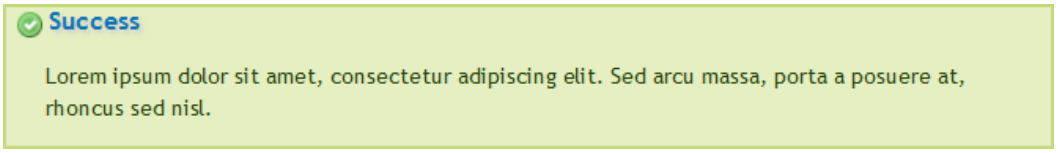
Help

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed arcu massa, porta a posuere at, rhoncus sed nisl.

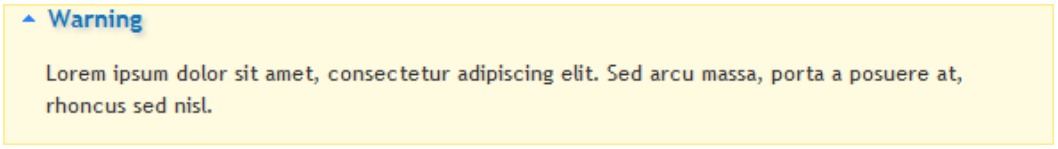
Information



Success



Warning



Examples for Message

Message has the following variations:

- **Default Functionality**. See above.
- **Dynamic Message**. You can dynamically alter the message displayed to the user by using the message control's Text and DisplayMode properties.

Default functionality example

.aspx

```
<ektronUI:Message ID="uxErrorMessage" DisplayMode="Error" runat="server">
    <ContentTemplate>An example of an <code>Error</code> message.</ContentTemplate>
</ektronUI:Message>

<ektronUI:Message ID="uxHelpMessage" DisplayMode="Help" runat="server">
    <ContentTemplate>An example of a <code>Help</code> message.</ContentTemplate>
</ektronUI:Message>

<ektronUI:Message ID="uxInformationMessage" DisplayMode="Information" runat="server">
    <ContentTemplate>An example of an <code>Information</code>
message.</ContentTemplate>
</ektronUI:Message>

<ektronUI:Message ID="uxSuccessMessage" DisplayMode="Success" runat="server">
    <ContentTemplate>An example of a <code>Success</code> message.</ContentTemplate>
</ektronUI:Message>

<ektronUI:Message ID="uxWarningMessage" DisplayMode="Warning" runat="server">
    <ContentTemplate>An example of a <code>Warning</code> message.<br />
```

```
    <br />
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent nec erat risus.
Suspendisse in varius tortor. Nunc at tellus diam, a sodales neque. Phasellus
condimentum hendrerit porta. Donec pretium leo et leo congue sodales. Etiam pretium
suscipit orci, a commodo diam tempus eget. Donec ac lacus ut augue imperdiet convallis
quis at diam. Nunc eget nunc neque, eget adipiscing velit. Donec venenatis egestas
sagittis. Duis mattis, eros at gravida venenatis, quam justo posuere urna, pharetra
semper tortor eros quis felis. Donec vitae porta ligula. Vivamus et vestibulum elit.
Nulla rhoncus ipsum vel nibh tristique tempus. In velit tellus, faucibus et commodo ut,
vulputate nec tellus. Mauris non est quis nisl fermentum egestas. Donec vestibulum
consequat odio et rhoncus.</ContentTemplate>
</ektronUI:Message>
```

NOTE: .aspx.cs not required.

Dynamic Message example

.aspx

```
<ektronUI:Message DisplayMode="Information" ID="uxMessage" runat="server">
    <ContentTemplate>
    To change the text and display mode shown in this message, select a different
display mode option from the drop down menu provided.
    </ContentTemplate>
</ektronUI:Message>
<ektronUI:Label ID="uxDisplayModeLabel" runat="server"
AssociatedControlID="aspDisplayMode" Text="Display Mode: " />
<asp:DropDownList id="aspDisplayMode" name="culture"  AutoPostBack="true"
OnSelectedIndexChanged="aspDisplayModeChange_change" runat="server"></asp:DropDownList>
```

.aspx.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Globalization;
using System.Web.UI.WebControls;
using Ektron.Site.Developer;
using Ektron.Cms.Framework.UI.Controls.EktronUI;

public partial class DyanmicMessage : SampleControlPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            foreach (int mode in System.Enum.GetValues(typeof(Message.DisplayModes)))
            {
                ListItem item = new ListItem();
                item.Value = mode.ToString();
                item.Text = System.Enum.GetName(typeof(Message.DisplayModes), mode);
                item.Selected = (aspDisplayMode.SelectedValue == mode.ToString()) ? true
: false;
                aspDisplayMode.Items.Add(item);
                item = null;
```

```
            }
        }
    }

    protected void aspDisplayModeChange_change(Object sender, EventArgs e)
    {
        int value;
        int.TryParse(aspDisplayMode.SelectedValue, out value);
        uxMessage.DisplayMode = (Message.DisplayModes)value;

        switch(uxMessage.DisplayMode) {
            case Message.DisplayModes.Error:
                uxMessage.Text = "An error occurred (not really, but we wanted to
demonstrate what one might look like to the user).";
                break;
            case Message.DisplayModes.Help:
                uxMessage.Text = "Provide tips and/or tricks to the user through helpful
messages like this one.";
                break;
            case Message.DisplayModes.Information:
                uxMessage.Text = "The select drop down below merely alters the
<code>Text</code> property of the message to provide different messages with each new
choice made below.";
                break;
            case Message.DisplayModes.Success:
                uxMessage.Text = "Let the user know when they have successfully
compelted steps in your forms.";
                break;
            case Message.DisplayModes.Warning:
                uxMessage.Text = "Caution users against certain actions or inform them
of possible consequences for choices they make within your application.";
                break;
        }
    }

    #region helpers
    public void SortDropDown(DropDownList cbo)
    {
        List<ListItem> lstListItems = new List<ListItem>();
        foreach (ListItem li in cbo.Items)
        {
            lstListItems.Add(li);
        }
        lstListItems.Sort(new ListItemComparer());
        cbo.Items.Clear();
        cbo.Items.AddRange(lstListItems.ToArray());
    }


    public class ListItemComparer : IComparer<ListItem>
    {

        public int Compare(ListItem x, ListItem y)
        {
            CaseInsensitiveComparer c = new CaseInsensitiveComparer();
            return c.Compare(x.Text, y.Text);
```

```
        }
    }
    #endregion
}
```

Pager

8.50 and higher

```
<EktronUI:Pager>
```

Limits the number of items on a page and provide simple navigation to other items. Use this with any controller that supports `IPageable`.

NOTE: You can improve performance by setting the `ResultsPerPage` property so this request is on the server side as the data is aggregated.

The default template used by the `<EktronUI:Pager>` server control is located at:

```
[site root]/Workarea/FrameworkUI/Templates/EktronUI/Pager.ascx
```

Attributes for Pager

Events for Pager

None.

Methods for Pager

None.

Theming for Pager

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div id="uxPager" class="ektron-ui-control ektron-ui-pager">
  <span class="previous alignLeft">
    <span class="ektron-ui-button">
      <a href="#Previous"
        onclick="if ($ektron(this).is('.ui-state-disabled,
        .ui-state-disabled &gt; span')){ return false;};
        WebForm_DoPostBackWithOptions(new WebForm_PostBackOptions
        ('pager1$ctl00$Pager$uxPrevious', '', true, '', '', false, true));"
        title="" name="pager1_Pager_0_uxPrevious_0_Button"
```

```
        id="pager1_Pager_0_uxPrevious_0_Button"
        class="ui-button ui-widget ui-state-default ui-corner-all
          ui-button-disabled ui-state-disabled ui-button-text-icon-primary"
        role="button" aria-disabled="true" disabled="disabled">
        <span class="ui-button-icon-primary ui-icon ui-icon-seek-prev"
          onclick="if ($ektron(this).is('.ui-state-disabled,
          .ui-state-disabled &gt; span')){ return false;};
          WebForm_DoPostBackWithOptions(new WebForm_PostBackOptions
          ('pager1$ctl00$Pager$uxPrevious', '', true, '', '', false,
            true)); return false;">
        </span>
        <span class="ui-button-text" onclick="if ($ektron(this).is
          ('.ui-state-disabled, .ui-state-disabled &gt; span'))
            { return false;};
          WebForm_DoPostBackWithOptions(new WebForm_PostBackOptions
          ('pager1$ctl00$Pager$uxPrevious', '', true, '', '', false,
            true)); return false;">Previous
        </span>
      </a>
    </span>
    <script type="text/javascript"
      id="pager1_Pager_0_uxPrevious_0_pager1_Pager_0_uxPrevious_
        0EktronScriptBlockdltgg_0">
      <!--//-->
      <![CDATA[//>
      <!--
      Ektron.ready(function(event, eventName)
      {
      $ektron("#pager1_Pager_0_uxPrevious_0_Button").button(
        {
        "disabled":true,"icons":{"primary":"ui-icon-seek-prev"},
        "create":    function(event, ui)
          {
          var link = $ektron(event.target);
          var click = link[0].getAttribute('onclick');
          link.children('span').each(function(index)
            {
            this.setAttribute('onclick', click + ' return false;');
            return true;
            });
          }
        });
      });
      //-->
      <!]]>
    </script>
  </span>
  …
  <input type="hidden" value="0"
    id="pager1_Pager_0_aspCurrentPage_0"
    name="pager1$ctl00$Pager$aspCurrentPage">
</div>
```

Example for Pager

Default functionality example

.aspx

```
<ektron:SiteSearchInputView ControllerID="SearchController1" ID="SiteSearch1"
runat="server"></ektron:SiteSearchInputView>
<ektron:SiteSearchController ID="SearchController1" runat="server" />
<ektron:SiteSearchResultsView ControllerID="SearchController1" ID="SearchResults1"
runat="server"></ektron:SiteSearchResultsView>
<ektronUI:Pager ID="pager1" runat="server" PageableControlID="SearchController1"
ResultsPerPage="5"></ektronUI:Pager>
```

NOTE: .aspx.cs not required.

Tabs

8.50 and higher

```
<EktronUI:Tabs>
```

Provides horizontally tabbed headers to display multiple sections of content. (See also *Accordion* on page 1908.)

When you click a tab, it is highlighted and the content for that section appears.

Events for Tabs

- `Click`. Set to server side event handler that is called when a user clicks the tab. Must be set on the contained Tab control.
- `Command`. The Command event is raised when the control is clicked. By associating a command name with the control, one method can handle the event from multiple controls, and you can programmatically determine the specific control that triggered the event by examining the event's command name.

Methods for Tabs

- `SetActiveTab(EktronUI.Tab)`. Selects the supplied tab; must be a child of the `<EktronUI:Tabs>` control.
- `SetActiveTab(string)`. Selects the child tab that has the supplied name.

Theming for Tabs

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div class="ektron-ui-control ektron-ui-accordion
  devsite-demo-tabs ui-tabs ui-widget
```

```
  ui-widget-content ui-corner-all"
  id="uxDemoTabs">
  <ul class="ui-tabs-nav ui-helper-reset
    ui-helper-clearfix ui-widget-header
    ui-corner-all">
    <li class="ui-state-default ui-corner-top
      ui-tabs-selected ui-state-active">
      <a title="Tab One" href="#uxDemoTabs_uxTab1">
        Tab One
      </a>
    </li>
    <li class="ui-state-default ui-corner-top">
      <a title="Tab Two" href="#uxDemoTabs_uxTab2">
        Tab Two
      </a>
    </li>
    <li class="ui-state-default ui-corner-top">
      <a title="Tab Three"
        href="#uxDemoTabs_uxTab3">
        Tab Three
      </a>
    </li>
  </ul>
  <div class="ektron-ui-control ektron-ui-tab
    EktronUI ui-tabs-panel ui-widget-content
    ui-corner-bottom" id="uxDemoTabs_uxTab1">
    <h4>Inside tab one</h4>
      Lorem ipsum dolor sit amet, consectetur
      adipisicing elit, sed do eiusmod tempor
      incididunt ut labore et dolore magna aliqua.
  </div>
  <div class="ektron-ui-control ektron-ui-tab
    EktronUI ui-tabs-panel ui-widget-content
    ui-corner-bottom ui-tabs-hide"
    id="uxDemoTabs_uxTab2">
    <h4>Inside tab two</h4>
      Ut enim ad minim veniam, quis nostrud
      exercitation ullamco laboris nisi ut aliquip
      ex ea commodo consequat. Duis aute irure dolor
      in reprehenderit in voluptate velit esse
      cillum dolore eu fugiat nulla pariatur.
  </div>
  <div class="ektron-ui-control ektron-ui-tab
    EktronUI ui-tabs-panel ui-widget-content
    ui-corner-bottom ui-tabs-hide"
    id="uxDemoTabs_uxTab3">
    <h4>Inside tab three</h4>
      Excepteur sint occaecat cupidatat non
      proident, sunt in culpa qui officia deserunt
      mollit anim id est laborum.
  </div>
  <input type="hidden" value="0"
    id="uxDemoTabs_TabsSelectedIndex"
    name="uxDemoTabs$TabsSelectedIndex" />
</div>
```

Examples for Tabs

Tabs has the following variations:

- **Default Functionality**. Horizontal tab controls display grouped content.
- **Active Tab**. Shows the active tab that is set in the code-behind in the page's OnLoad method.
- **Contains Controls**. Clicking on a tab fills a field with postback text. AutoHeight is disabled, which causes each tab to fit their contents rather than adjusting all tabs to the size of the largest.
- **Databind to Add Tabs**. Adds a tab by Databinding.
- **Dynamically Add Tabs**. Adds a tab programatically.
- **Postback**. Displays text that is associated with the click event. The Tabs control contains a server-side onclick event handler that generates a postback when clicked.

Default functionality example

.aspx

```
<ektronUI:Tabs ID="uxDemoTabs" runat="server" CssClass="devsite-demo-tabs" >
    <ektronUI:Tab ID="uxTab1" runat="server" CssClass="EktronUI" Text="Tab One" >
        <ContentTemplate>
            <h4>Inside tab one</h4>
            Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.
        </ContentTemplate>
    </ektronUI:Tab>
    <ektronUI:Tab ID="uxTab2" runat="server" CssClass="EktronUI" Text="Tab Two" >
        <ContentTemplate>
            <h4>Inside tab two</h4>
            Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate
velit esse cillum dolore eu fugiat nulla pariatur.
        </ContentTemplate>
    </ektronUI:Tab>
    <ektronUI:Tab ID="uxTab3" runat="server" CssClass="EktronUI" Text="Tab Three" >
        <ContentTemplate>
            <h4>Inside tab three</h4>
            Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
deserunt mollit anim id est laborum.
        </ContentTemplate>
    </ektronUI:Tab>
</ektronUI:Tabs>
```

NOTE: .aspx.cs not required.

Active Tab example

.aspx

```
<ektronUI:Tabs ID="uxDemoTabs" runat="server" >
    <ektronUI:Tab ID="uxTab1" runat="server" CssClass="EktronUI" Text="Tab One" >
        <ContentTemplate>
            <h2>Inside tab one</h2>
            <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua.</p>
        </ContentTemplate>
    </ektronUI:Tab>
    <ektronUI:Tab ID="uxTab2" runat="server" CssClass="EktronUI" Text="Tab Two" >
        <ContentTemplate>
            <h2>Inside tab two</h2>
            <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi
ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate
velit esse cillum dolore eu fugiat nulla pariatur.</p>
        </ContentTemplate>
    </ektronUI:Tab>
    <ektronUI:Tab ID="uxTab3" runat="server" CssClass="EktronUI" Text="Tab Three" >
        <ContentTemplate>
            <h2>Inside tab three</h2>
            <p>Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
deserunt mollit anim id est laborum.</p>
        </ContentTemplate>
    </ektronUI:Tab>
</ektronUI:Tabs>
```

.aspx.cs

```
protected override void OnLoad(EventArgs e)
{
    base.OnLoad(e);
    uxDemoTabs.SetActiveTab(uxTab2);
}
```

Contains Controls example

.aspx

```
<ektronUI:Tabs ID="uxDemoTabs" runat="server" >
    <ektronUI:Tab ID="uxTab1" runat="server" CssClass="EktronUI" Text="Tab One" >
        <ContentTemplate>
            <ektronUI:TextField ID="uxTextField" runat="server" Text="" />
            <ektronUI:ButtonSet ID="ButtonSet1" runat="server">
                    <ektronUI:Button ID="Button1" runat="server"  DisplayMode="Checkbox"
OnClick="Button1_click" />
                    <ektronUI:Button ID="Button2" runat="server"  DisplayMode="Checkbox"
OnClick="Button2_click" />
                    <ektronUI:Button ID="Button3" runat="server"  DisplayMode="Checkbox"
OnClick="Button3_click" />
                </ektronUI:ButtonSet>
        </ContentTemplate>
    </ektronUI:Tab>
```

```
    <ektronUI:Tab ID="uxTab2" runat="server" CssClass="EktronUI" Text="Tab Two" >
        <ContentTemplate>
            <cms:Login ID="login1" runat="server" /><br /><br />
            <cms:ContentBlock ID="cb1" runat="server" DefaultContentID="30" />
            <cms:ListSummary ID="ls1" runat="server" FolderID="78" Recursive="false" />
        </ContentTemplate>
    </ektronUI:Tab>
</ektronUI:Tabs>
```

.aspx.cs

```
protected override void OnLoad(EventArgs e)
{
    base.OnLoad(e);
}

 protected void Button1_click(object sender, EventArgs e)
    {
        uxTextField.Text = "CheckBox one clicked!";
    }

    protected void Button2_click(object sender, EventArgs e)
    {
        uxTextField.Text = "CheckBox two clicked!";
    }

    protected void Button3_click(object sender, EventArgs e)
    {
        uxTextField.Text = "CheckBox three clicked!";
    }
```

Databind to Add Tabs example

.aspx

```
<ektronUI:Tabs ID="uxTabsControl" runat="server" CssClass="class_tabs1"
ViewStateMode="Enabled" >
        <ektronUI:Tab ID="Tab0" runat="server" Text="Tab 0" CssClass="class_tab0">
            <ContentTemplate>
                Tab-Zero (declaratively added)
            </ContentTemplate>
        </ektronUI:Tab>
    </ektronUI:Tabs>
```

.aspx.cs

```
protected void Page_Init(object sender, EventArgs e) {
        this.SelectedTab = "uxEktronUiTab";
        this.SelectedAccordion = "uxTabsTab";
        this.SelectedSample = "ActiveTab";
        this.ComponentName = "EktronUI:Tabs";
        this.ComponentUrl = "/Framework/UI/Controls/EktronUI/Tabs/ActiveTab.aspx";
        this.IsDemoSafe = true;
        this.DocumentationName = "Ektron.Cms.Framework.UI.Controls.EktronUI.Tabs";
        uxSummary.BaseClasses.Add
```

```
("Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets.TabsBase");

        LoadTabControls();
    }

    protected void LoadTabControls() {

        var tabs = new TabData[]{
            new TabData(){Text = "Tab 1", ContentText="tab one (added by DataBinding)",
Selected = true},
            new TabData(){Text = "Tab 2", ContentText="tab two (added by DataBinding)"},
            new TabData(){Text = "Tab 3", ContentText="tab three (added by
DataBinding)"}
        };

        uxTabsControl.DataSource = tabs;
        uxTabsControl.DataBind();
    }
```

Dynamically Add Tabs example

.aspx

```
<ektronUI:Tabs ID="uxTabsControl" runat="server" CssClass="class_tabs1">
            <ektronUI:Tab ID="Tab1" runat="server" Text="Tab1" >
                <ContentTemplate>
                    Tab1 (declaratively added)
                </ContentTemplate>
            </ektronUI:Tab>
            <ektronUI:Tab ID="Tab2" runat="server" Text="Tab2" >
                <ContentTemplate>
                    Tab2 (declaratively added)
                </ContentTemplate>
            </ektronUI:Tab>
        </ektronUI:Tabs>
        <br />
    <asp:Literal runat="server" ID="messageLiteral" />
```

.aspx.cs

```
protected void Page_Init(object sender, EventArgs e) {
            this.SelectedTab = "uxEktronUiTab";
            this.SelectedAccordion = "uxTabsTab";
            this.SelectedSample = "ActiveTab";
            this.ComponentName = "EktronUI:Tabs";
            this.ComponentUrl = "/Framework/UI/Controls/EktronUI/Tabs/ActiveTab.aspx";
            this.IsDemoSafe = true;
            this.DocumentationName = "Ektron.Cms.Framework.UI.Controls.EktronUI.Tabs";
            uxSummary.BaseClasses.Add
("Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets.TabsBase");

            LoadTabControls();
        }

        protected void LoadTabControls() {
```

```
            var tab = new Tab { Text = "Dynamic Tab1", ID = "DynamicTab1", Content =
"Inside Dynamic Tab1 (added programatically)"};
            uxTabsControl.AddTab(tab);
            AddChildButton(tab);

            tab = new Tab {Text = "Dynamic Tab2", ID = "DynamicTab2", Content = "Inside
Dynamic Tab2 (added programatically)"};
            uxTabsControl.AddTab(tab);
        }

        private void AddChildButton(Tab tab) {
            tab.Controls.Add(new LiteralControl(""));
            var button = new Ektron.Cms.Framework.UI.Controls.EktronUI.Button();
            button.Text = "A button child control";
            button.ID = "Button1";
            button.OnClientClick = "alert('button clicked!')";
            button.Click += OnButtonClicked;
            tab.Controls.Add(button);
        }

        protected void OnButtonClicked(object source, EventArgs args) {
            uxTabsControl.SetActiveTab("DynamicTab1");
            messageLiteral.Text = "Button Clicked!";
        }
```

Postback example

.aspx

```
<ektronUI:Tabs ID="uxDemoTabs" runat="server" >
    <ektronUI:Tab ID="uxTab1" runat="server" CssClass="EktronUI" Text="Tab One" >
        <ContentTemplate>
            Static content... (click the other tab)
        </ContentTemplate>
    </ektronUI:Tab>
    <ektronUI:Tab ID="uxTab2" runat="server" CssClass="EktronUI" Text="Tab Two"
OnClick="uxTab2_click" >
        <ContentTemplate>
            <b>
                <asp:Literal ID="literal1" runat="server" />
            </b>
        </ContentTemplate>
    </ektronUI:Tab>
</ektronUI:Tabs>
```

.aspx.cs

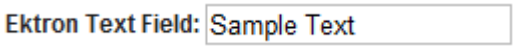
```
protected void uxTab2_click(object sender, EventArgs e)
{
    uxTab2.Text = "Tab Two (been clicked)";
    uxDemoTabs.SetActiveTab(uxTab2);
    literal1.Text = "Content set in code-behind, during click event servicing...)";
}
```

TextField

8.50 and higher

```
<EktronUI:TextField>
```

Creates and input field in which you can enter text.



The `<EktronUI:TextField>` control is a non-data-typed I/O control. Input is optional, but you can make it mandatory by adding a *RequiredRule* to the control's *ValidationRules*. The *Text* property is the primary means to query or update the field.

Events for TextField

None.

Methods for TextField

None.

Theming for TextField

Styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<span class="ektron-ui-control ektron-ui-input
  ektron-ui-textField"
  id="uxTextField_TextField_uxTextField">
  <input type="text"
    id="uxTextField_TextField_aspInput"
    value="Sample Text"
    name="uxTextField$TextField$aspInput">
</span>
```

Examples for TextField

TextField has the following variations:

- **Default Functionality**. "Sample Text" is placed in the field that you can replace with any text. Click **Submit** to read back the text on postback.

- **Custom Validation**. CustomRuleValidator is added to the TextField input control to verify that the user input contains a specified word. If you remove the specified word from the text field and click **Submit**, a message appears saying that the user must have the specified word in the text field. You can re-use the custom validation method for more than one rule, or have a unique method for each.

  The custom rule validator has a parameter of type `CustomRuleValidatorEventArgs`. This type is defined in `Ektron.Cms.Framework.UI.Controls.EktronUI.Core.Validation`.
- **Disabled TextField**. "Sample text" is placed in the field and can be seen but not modified. Validation is not needed for a disabled (or invisible) control.
- **Regex Validation**. Validates specific data, such as a street address (`123 Demo Road`) using `ServerRegex="\d+\s+\w+"`. Replace `123 Demo Road` with `Sample Text` and click **Submit** to see a message that shows that the new input does not meet the requirements of the RegEx rule. Enter `1600 Pennsylvania Avenue` and click **Submit** to cause `Page.IsValid` to return `True`. The example specifies a server-side validation RegEx only; client-side regular expression validators are specified using ClientRegex. Because a server-side RegEx is specified, the form submits and any related validation failures are captured on the server side.
- **Validation**. Adds RequiredRule to the TextField input control so that the field changes color if blank is entered, indicating that the user must enter text in the field. The client-side validator prevents a postback from taking place, and displays the supplied ErrorMessage text to the user.
- **Validation No-Client**. Adds RequiredRule to the TextField input control and disables client side validation by declaratively setting `ClientValidationEnabled="false"` on the rule. Because client-side validation is disabled, the form sends a postback to the server (even with invalid data). Valid input values are tested using `Page.IsValid` in the button-click event handler and, for example, only store user values to the database if `Page.IsValid` returns `True`. Server-side validation is performed automatically by ASP.NET, but you also can validate it programmatically by calling `Page.Validate()`.
- **Validation Groups**. Organizes content into sections, and validates only the relevant section. For example, you may have a user login section display on the same form as a user registration section. You can assign ValidationGroup properties to the login related controls, and a different one for the registration controls. Clicking the submit buttons with any text causes `Page.IsValid` to return `True`.

Default functionality example

.aspx

```
<ektronUI:Label id="uxTextFieldLabel" AssociatedControlID="uxTextField"
runat="server">Ektron Text Field:</ektronUI:Label>
<ektronUI:TextField ID="uxTextField" runat="server" Text="Sample Text">
```

```
</ektronUI:TextField><br />
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button>
```

.aspx.cs

```
protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // get the textfield value
    string value = uxTextField.Text;
}
```

Custom Validation example

.aspx

```
<ektronUI:Label id="uxTextFieldLabel" AssociatedControlID="uxTextField"
runat="server">Ektron Text Field:</ektronUI:Label>
<ektronUI:TextField ID="uxTextField" runat="server" Text="Custom validator demo" >
    <ValidationRules>
        <ektronUI:CustomRule ServerValidationEnabled="true"
OnValidating="MyCustomRuleValidator" />
    </ValidationRules>
</ektronUI:TextField><br />
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button><br />
<asp:Literal ID="InfoMessage" runat="server" />
```

.aspx.cs

```
protected void MyCustomRuleValidator(Object sender, CustomRuleValidatorEventArgs e)
{
    if (e.Enabled)
    {
        string userText = string.Empty;

        var textBoxControl = e.ControlToValidate as System.Web.UI.WebControls.TextBox;
        if (textBoxControl != null)
        {
            userText = textBoxControl.Text;
        }

        e.IsValid = userText.ToLower().Contains("custom");
        e.ErrorMessage = "Input must contain the word 'custom'";
    }
    else
    {
        e.IsValid = true;
    }
}

protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // if valid, then get the textfield value
    if (Page.IsValid)
```

```
    {
        string value = uxTextField.Text;
    }

    ShowInfo();
}

protected void ShowInfo()
{
    if (Page.IsValid)
    {
        InfoMessage.Text = "Page.IsValid: True";
        return;
    }

    // get the error message of the first failing validator
    foreach (IValidator validator in Page.Validators)
    {
        if (!validator.IsValid)
        {
            InfoMessage.Text = "Page.IsValid: False - " + validator.ErrorMessage;
            break;
        }
    }
}
```

Disabled TextField example

.aspx

```
<ektronUI:Label id="uxTextFieldLabel" AssociatedControlID="uxTextField"
runat="server">Ektron Text Field:</ektronUI:Label>
<ektronUI:TextField ID="uxTextField" runat="server" Enabled="false" />
```

.aspx.cs

```
protected void Page_Init(object sender, EventArgs e)
{
    // set the textfield value
    uxTextField.Text = "Sample Text";
}
```

Regex Validation example

.aspx

```
<ektronUI:Label id="uxTextFieldLabel" AssociatedControlID="uxTextField"
runat="server">Ektron Text Field:</ektronUI:Label>
<ektronUI:TextField ID="uxTextField" runat="server" Text="123 Demo Road" >
    <ValidationRules>
        <ektronUI:RegexRule ServerRegex="\d+\s+\w+" ErrorMessage="Input does not satisfy
the Regex-Rule!" />
    </ValidationRules>
</ektronUI:TextField><br />
```

```
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button><br />
<asp:Literal ID="InfoMessage" runat="server" />
```

.aspx.cs

```
protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // if valid, then get the textfield value
    if (Page.IsValid)
    {
        string value = uxTextField.Text;
    }

    ShowInfo();
}

protected void ShowInfo()
{
    if (Page.IsValid)
    {
        InfoMessage.Text = "Page.IsValid: True";
        return;
    }

    // get the error message of the first failing validator
    foreach (IValidator validator in Page.Validators)
    {
        if (!validator.IsValid)
        {
            InfoMessage.Text = "Page.IsValid: False - " + validator.ErrorMessage;
            break;
        }
    }
}
```

Validation example

.aspx

```
<ektronUI:Label id="uxTextFieldLabel" AssociatedControlID="uxTextField"
runat="server">Ektron Text Field:</ektronUI:Label>
<ektronUI:TextField ID="uxTextField" runat="server" Text="Sample Text">
    <ValidationRules>
        <ektronUI:RequiredRule ErrorMessage="Text is required!" />
    </ValidationRules>
</ektronUI:TextField>
<ektronUI:ValidationMessage ID="uxValidationMessage" runat="server"
AssociatedControlID="uxTextField" /><br />
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button>
```

.aspx.cs

```
protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // if input is valid, then get the textfield value
    if (Page.IsValid)
    {
        string value = uxTextField.Text;
        // at this point we could process the user input, store it to a database, etc...
    }
}
```

Validation No-Client example

.aspx

```
<ektronUI:Label id="uxTextFieldLabel" AssociatedControlID="uxTextField"
runat="server">Ektron Text Field:</ektronUI:Label>
<ektronUI:TextField ID="uxTextField" runat="server" Text="Sample Text">
    <ValidationRules>
        <ektronUI:RequiredRule ErrorMessage="Value is required!"
ClientValidationEnabled="false" />
    </ValidationRules>
</ektronUI:TextField><br />
<ektronUI:Button ID="uxSubmitButton" OnClick="uxSubmitButton_click" runat="server"
Text="Submit"></ektronUI:Button><br />
<asp:Literal ID="InfoMessage" runat="server" />
```

.aspx.cs

```
protected void uxSubmitButton_click(object sender, EventArgs e)
{
    // if valid, then get the textfield value
    if (Page.IsValid)
    {
        string value = uxTextField.Text;
    }

    ShowInfo();
}

protected void ShowInfo()
{
    if (Page.IsValid)
    {
        InfoMessage.Text = "Page.IsValid: True";
        return;
    }

    // get the error message of the first failing validator
    foreach (IValidator validator in Page.Validators)
    {
        if (!validator.IsValid)
        {
            InfoMessage.Text = "Page.IsValid: False - " + validator.ErrorMessage;
```

```
            break;
        }
    }
}
```

Validation Groups example

.aspx

```
<ektronUI:Label id="uxTextFieldLabel" AssociatedControlID="uxTextFieldOne"
runat="server">Ektron Text Field:</ektronUI:Label>
    <ektronUI:TextField ID="uxTextFieldOne" ValidationGroup="Group1" runat="server"
Text="Sample Text">
        <ValidationRules>
            <ektronUI:RequiredRule ErrorMessage="Text is required!" />
        </ValidationRules>
    </ektronUI:TextField>
    <ektronUI:Button ID="uxSubmitButtonOne" OnClick="uxSubmitButtonOne_click"
ValidationGroup="Group1" runat="server" Text="Submit Group 1"></ektronUI:Button>
    <ektronUI:ValidationMessage ID="uxValidationMessage" runat="server"
AssociatedControlID="uxTextFieldOne" /><br />
    <ektronUI:Label id="Label1" AssociatedControlID="uxTextFieldTwo"
runat="server">Ektron Text Field:</ektronUI:Label>
    <ektronUI:TextField ID="uxTextFieldTwo" ValidationGroup="Group2" runat="server"
Text="Sample Text">
        <ValidationRules>
            <ektronUI:RequiredRule ErrorMessage="Text is required!" />
        </ValidationRules>
    </ektronUI:TextField>
    <ektronUI:Button ID="uxSubmitButtonTwo" OnClick="uxSubmitButtonTwo_click"
ValidationGroup="Group2" runat="server" Text="Submit Group 2"></ektronUI:Button>
    <ektronUI:ValidationMessage ID="ValidationMessage1" runat="server"
AssociatedControlID="uxTextFieldTwo" /><br />
    <asp:Literal ID="InfoMessage" runat="server" />
```

.aspx.cs

```
protected void uxSubmitButtonOne_click(object sender, EventArgs e)
{
    // get the validated textfield value for group one
    if (Page.IsValid)
    {
        string value = uxTextFieldOne.Text;
    }

    ShowInfo();
}

protected void uxSubmitButtonTwo_click(object sender, EventArgs e)
{
    // get the validated textfield value for group two
    if (Page.IsValid)
    {
        string value = uxTextFieldTwo.Text;
    }
```

```
    ShowInfo();
}

protected void ShowInfo()
{
    if (Page.IsValid)
    {
        InfoMessage.Text = "Page.IsValid: True";
        return;
    }

    // get the error message of the first failing validator
    foreach (IValidator validator in Page.Validators)
    {
        if (!validator.IsValid)
        {
            InfoMessage.Text = "Page.IsValid: False - " + validator.ErrorMessage;
            break;
        }
    }
}
```

Tree

9.00 and higher

```
<EktronUI:Tree>
```

A generic tree control renders items that are added declaratively or programatically as a nested unordered-list.

The markup is generated (on the client-side by default) using a template that provides CSS hooks for styling. The template can be customized to generate different markup if needed.

Part of a family of controls; ranging from the generic "Tree" and "ContextMenuTree" controls (which require their data to be supplied either declaratively or programatically), to controls that are intended to work with a particular Framework UI and it's data (pulling the data automatically); FolderTree, MenuTree and TaxonomyTree.

All of these controls were designed to be relatively simple to use, yet they provide many ways to customize their behavior as well as the resulting look and feel - maximizing flexibility for the end developer.

Events for Tree

- `Callback`. Fires when an Ajax callback is made from the tree controls client-side code (JavaScript). Values may be inspected, modified, or the default processing can be aborted - resulting in the current/custom values being serialized back to the client.

  Parameter Type: `TreeAjaxCallbackEventArgs`.

- `Click`. Event fires when a tree node is clicked.

  Parameter Type: `EventArgs`.

- `Command`. Command event will fire when a node with a CommandName is clicked. If CommandArgument is specified on the node, it will also be passed to the handler.

  Parameter Type: `CommandEventArgs`.

- `DataBinding`. Fires when the control is data binding.
- `ItemDataBound`. Fires for each node when the control is databinding.

  Parameter Type: `TreeItemDataBoundEventArgs<T>`.

- `PreSerializeData`. Event fires just before serializing the data for sending to the client. This is the last opportunity to inspect or modify the data before it is made available for rendering the UI.

  Parameter Type: `EventArgs`.

- `SelectionEvent`. Fired for all postbacks, passes the IDs of the currently

selected nodes.

Parameter Type: `TreeSelectionEventArgs`.

Methods for Tree

- `AddProcessor(ITreeDataProcessor)`. Provides an extension point, to add a custom ITreeDataProcessor adapter (`Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets.ITreeDataProcessor`) to be inserted into the tree-data processing chain.
- `DataBind()`. Binds the control to the current data source (specified using the DataSource property).
- `GetNode(string)`. Returns the ID-specified node, if found.

  Return Type: `Ektron.Cms.Framework.UI.Tree.ITreeNode`.
- `GetNodeFromPath(string)`. Returns the path-specified node, if found.

  Return Type: `Ektron.Cms.Framework.UI.Tree.ITreeNode`.
- `GetParentNode(string)`. Returns the parent of the ID-specified node, if found.

  Return Type: `Ektron.Cms.Framework.UI.Tree.ITreeNode`.

Theming for Tree

Styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div class="ektron-ui-control ektron-ui-tree ektron-ui-genericTree">
  <ul class="ektron-ui-tree-root">
    <li class="ektron-ui-tree-leaf ektron-ui-tree-generic">
      <div class="hitLocation">
        <div class="selectionLocation none">
          <span class="ui-icon"></span>
          <span class="textWrapper clickable" onclick="alert('node clicked!');; return
false;">Item One (added in code-behind) </span>
        </div>
      </div>
    </li>
    <li class="ektron-ui-tree-leaf ektron-ui-tree-generic">
      <div class="hitLocation">
        <div class="selectionLocation none">
          <span class="ui-icon"></span>
          <span class="textWrapper clickable" onclick="alert('node clicked!');; return
false;">Item Two (added in code-behind) </span>
        </div>
      </div>
```

```
    </li>
    <li class="ektron-ui-tree-leaf ektron-ui-tree-generic">
      <div class="hitLocation">
        <div class="selectionLocation none">
          <span class="ui-icon"></span>
          <span class="textWrapper clickable" onclick="alert('node clicked!');; return
false;">Item Three (added in code-behind) </span>
        </div>
      </div>
    </li>
    <li class="ektron-ui-tree-leaf ektron-ui-tree-generic ektron-ui-tree-type-
declarativedataleaf">
      <div class="hitLocation">
        <div class="selectionLocation none">
          <span class="ui-icon"></span>
          <span class="textWrapper clickable" onclick="alert('node clicked!');; return
false;">Declarative Item One </span>
        </div>
      </div>
    </li>
    <li class="ektron-ui-tree-branch ektron-ui-tree-generic ektron-ui-tree-type-
declarativedatabranch expandable">
      <div class="hitLocation">
        <div class="selectionLocation none">
          <span class="ui-icon"></span>
          <span class="textWrapper clickable" onclick="alert('node clicked!');; return
false;">Declarative Item Two </span>
        </div>
      </div>
      <ul>
        <li class="ektron-uiektron-ui-tree-leaf ektron-ui-tree-generic ektron-ui-tree-
type-declarativedataleaf">
          <div class="hitLocation">
            <div class="selectionLocation none">
              <span class="ui-icon"></span>
              <span class="textWrapper clickable" onclick="alert('node clicked!');;
return false;">Declarative Item Two Child-A </span>
            </div>
          </div>
        </li>
        <li class="ektron-ui-tree-leaf ektron-ui-tree-generic ektron-ui-tree-type-
declarativedataleaf">
          <div class="hitLocation">
            <div class="selectionLocation none">
              <span class="ui-icon"></span>
              <span class="textWrapper clickable" onclick="alert('node clicked!');;
return false;">Declarative Item Two Child-B </span>
            </div>
          </div>
        </li>
        <li class="ektron-ui-tree-leaf ektron-ui-tree-generic ektron-ui-tree-type-
declarativedataleaf">
          <div class="hitLocation">
            <div class="selectionLocation none">
              <span class="ui-icon"></span>
              <span class="textWrapper clickable" onclick="alert('node clicked!');;
```

```
return false;">Declarative Item Two Child-C </span>
            </div>
          </div>
        </li>
      </ul>
    </li>
  </ul>
</div>
```

Generic Tree examples

- **Tree**. Adds tree nodes declaratively and in code-behind.
- **Tree Bubble-Command**. Node-events bubble up to the tree controls parent-container, and handled in code-behind:
- **Tree Command-Click**. Wires up each node to cause server-side code to execute, using either a general click-driven-postback and using command names and arguments.
- **TreeCurrentNode**. Handles the tree-controls click event on the server (how to determine the current node ID), and using the auto-postback feature to cause the server side code to execute when a node is selected.
- **Tree Custom-Server-Template**. Alter the markup created by the tree control by binding the data to server controls contained in custom user-controls
- **Tree Custom-Template**. Alter the markup created by the tree control by using custom templates (generated by a custom user-control). Templates should be compatible with the jQuery Templating Plugin.
- **Tree In Repeater**. Uses a tree-control within a repeater's ItemTemplate, using the data that's bound to the repeater.
- **Tree Multi-Select**. Allows more than one node to be selected at a time.
- **Tree Single-Select**. Configures the tree control to allow only a single node to be selected at any time.

Tree example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <ektronUI:Tree ID="tree1" runat="server" >
        <ektronUI:TreeItem Id="dec1" runat="server" Text="Declarative Item One"
OnClientClick="alert('node clicked!');" />
```

```
        <ektronUI:TreeItem Id="dec2" runat="server" Text="Declarative Item Two"
OnClientClick="alert('node clicked!');" >
            <ektronUI:TreeItem Id="dec2a" runat="server" Text="Declarative Item Two
Child-A" OnClientClick="alert('node clicked!');" />
            <ektronUI:TreeItem Id="dec2b" runat="server" Text="Declarative Item Two
Child-B" OnClientClick="alert('node clicked!');" />
            <ektronUI:TreeItem Id="dec2c" runat="server" Text="Declarative Item Two
Child-C" OnClientClick="alert('node clicked!');" />
        </ektronUI:TreeItem>
    </ektronUI:Tree>
    </form>
</body>
</html>
```

.aspx.cs

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Web;
    using System.Web.UI;
    using System.Web.UI.WebControls;

    public partial class TreeDemo : System.Web.UI.Page
    {
        protected override void OnLoad(EventArgs e) {
            base.OnLoad(e);

            if (!Page.IsPostBack) {
                var data = new List<Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode>
{
                    new Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode() {Id =
"cb1", Text = "Item One (added in code-behind)", OnClientClick = "alert('node
clicked!');" },
                    new Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode() {Id =
"cb2", Text = "Item Two (added in code-behind)", OnClientClick = "alert('node
clicked!');"},
                    new Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode() {Id =
"cb3", Text = "Item Three (added in code-behind)", OnClientClick = "alert('node
clicked!');"}
                };

                tree1.DataSource = data;
                tree1.DataBind();
            }
        }
    }
```

Tree bubble-command example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>
```

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <asp:Literal ID="InfoLiteral" runat="server" /><br />
    <asp:DataList ID="container" runat="server" >
        <ItemTemplate>
            <ektronUI:Tree ID="tree1" runat="server" >
                <ektronUI:TreeItem Id="item1" runat="server" Text="Command: Code-Blue"
CommandName="DoTask" CommandArgument="CodeBlue" />
                <ektronUI:TreeItem Id="item2" runat="server" Text="Command: Code-Red"
CommandName="DoTask" CommandArgument="CodeRed" />
                <ektronUI:TreeItem Id="item3" runat="server" Text="Command: Edit"
CommandName="Edit" CommandArgument="Edit" />
            </ektronUI:Tree>
        </ItemTemplate>
    </asp:DataList>
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web.UI.WebControls;

public partial class TreeDemo : System.Web.UI.Page
{
    private string _info;

    protected override void OnInit(EventArgs e)
    {
        base.OnInit(e);

        container.ItemCommand += HandleCommand;
        container.EditCommand += HandleEditCommand;
    }

    protected override void OnLoad(EventArgs e)
    {
        base.OnLoad(e);

        var list = new List<string>() { "something" };
        container.DataSource = list;
        container.DataBind();
    }

    protected void HandleCommand(object sender, CommandEventArgs e)
    {
        _info = "Command Event from Container, Name: " + e.CommandName + ", Argument: "
+ e.CommandArgument;
```

```
    }

    protected void HandleEditCommand(object sender, CommandEventArgs e)
    {
        _info = "Edit Command Event from Container, Name: " + e.CommandName + ",
Argument: " + e.CommandArgument;
    }

    protected override void OnPreRender(EventArgs e)
    {
        base.OnPreRender(e);
        InfoLiteral.Text = _info;
    }
}
```

Tree command-click example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <asp:Literal ID="InfoLiteral" runat="server" /><br />
    <ektronUI:Tree ID="tree1" runat="server" >
        <ektronUI:TreeItem Id="item1" runat="server" Text="Click Handled Server-Side"
ClickCausesPostback="true" />
        <ektronUI:TreeItem Id="item2" runat="server" Text="Command: Code-Blue"
CommandName="DoTask" CommandArgument="CodeBlue" />
        <ektronUI:TreeItem Id="item3" runat="server" Text="Command: Code-Red"
CommandName="DoTask" CommandArgument="CodeRed" />
    </ektronUI:Tree>
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;

public partial class TreeDemo : System.Web.UI.Page
{
    private string _info;

    protected override void OnLoad(EventArgs e)
```

```
    {
        base.OnLoad(e);
        tree1.Click += HandleTreePostback;
        tree1.Command += HandleCommand;
    }

    protected void HandleTreePostback(object sender, EventArgs e)
    {
        _info = "Click Event, from Tree";

        var iTreeControl = (sender as ITreeControl);
        if (iTreeControl != null)
        {
            var currentNode = iTreeControl.CurrentNode;
            if (currentNode != null)
            {
                _info += ". Selected Node Id: " + currentNode.Id;
            }
        }
    }

    protected void HandleCommand(object sender, CommandEventArgs e)
    {
        _info = "Command Event, Name: " + e.CommandName + ", Argument: " +
e.CommandArgument;
    }

    protected override void OnPreRender(EventArgs e)
    {
        base.OnPreRender(e);
        InfoLiteral.Text = _info;
    }
}
```

TreeCurrentNode example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager runat="server" ID="ScriptManager1" />
        <asp:UpdatePanel ID="up1" runat="server" ChildrenAsTriggers="True" >
            <ContentTemplate>
                <b><asp:Literal ID="InfoLiteral" runat="server" Text="Please click a
tree node" /></b><br /><br />
                <ektronUI:Tree ID="tree1" runat="server"
```

```
                        OnClick="OnClick"
                        AutoPostBackOnSelectionChanged="True"
                        SelectionMode="SingleForAll"
                    />
            </ContentTemplate>
        </asp:UpdatePanel>
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.UI.Tree;

public partial class TreeDemo : System.Web.UI.Page
{
    protected override void OnLoad(EventArgs e)
    {
        base.OnLoad(e);
        if (!Page.IsPostBack)
        {
            tree1.DataSource = MakeTreeData(3, 5);
            tree1.DataBind();
        }
    }

    protected void OnClick(object sender, EventArgs e)
    {
        InfoLiteral.Text = "Current Node Id: " + tree1.CurrentNodeId;
    }

    #region tree-data helpers
    private List<Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode> MakeTreeData(int
numberOfRootFolders, int numberOfChildrenPerFolder)
    {
        var data = new List<Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode>();
        for (var rootCounter = 0; rootCounter < numberOfRootFolders; rootCounter++)
        {
            var rootNode = MakeNode(rootCounter);
            data.Add(rootNode);
            for (var childCounter = 0; childCounter < numberOfChildrenPerFolder;
childCounter++)
            {
                MakeNode(childCounter, rootNode);
            }
        }
        return data;
    }

    private static FolderTreeNode MakeNode(int index, ITreeNode parentNode = null)
    {
        var node = new FolderTreeNode();
        node.CanHaveChildren = true;
```

```
        if (parentNode != null)
        {
            node.Id = parentNode.Id + "_Child" + index.ToString();
            parentNode.HasChildren = true;
            (parentNode.Items ?? (parentNode.Items = new
Ektron.Cms.Framework.UI.Tree.TreeNodeCollection())).Add(node);
        }
        else
        {
            node.Id = "Folder" + index.ToString();
        }
        node.Text = node.Id;
        return node;
    }
    #endregion

}
```

Tree custom-server-template example

TreeDemo.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <b><asp:Literal ID="InfoLiteral" runat="server" Text="please click a tree node"
/></b><br /><br />
    <ektronUI:Tree ID="tree1" runat="server"
        TemplatePath="CustomServerTreeTemplate.ascx"
        OnCommand="OnCommand"
        />
    <style type="text/css">
        .customTextWrapper.selected { font-weight: bold;color: red; }
    </style>
    </form>
</body>
</html>
```

CustomServerTreeTemplate.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="CustomServerTreeTemplate.ascx.cs" Inherits="CustomServerTreeTemplate" %>
<div id="<%= ClientID %>" <%= CssManager %> >
    <asp:PlaceHolder ID="placeHolder" runat="server" />
    <ul id="<%= ClientID %>_TreeRootElement" class="ektron-ui-tree-root" >
        <asp:PlaceHolder ID="childContainerPlaceHolder" runat="server" />
```

```
    </ul>
</div>
```

CustomServerTreeItemTemplate.ascx

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="CustomServerTreeItemTemplate.ascx.cs" Inherits="CustomServerTreeItemTemplate"
%>
<asp:Repeater ID="nodeRepeater" runat="server" OnItemDataBound="HandleItemDataBound">
    <ItemTemplate>
        <li data-ektron-id="<%# Eval("Id") %>"
            <asp:Literal id="commandName" runat="server" /> <asp:Literal
id="commandArgument" runat="server" />
            class="customNode"
        >
            <span id="noUrlSpan" runat="server" visible="false" />
            <ul id="childContainer" runat="server" visible="false" />
        </li>
    </ItemTemplate>
</asp:Repeater>
```

TreeDemo.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.UI.Tree;
using TreeNode = Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode;

public partial class TreeDemo : System.Web.UI.Page
{
    protected override void OnInit(EventArgs e)
    {
        base.OnInit(e);
        tree1.DataSource = MakeTreeData(5, 3);
        tree1.DataBind();
    }

    protected void OnCommand(object sender, CommandEventArgs e)
    {
        InfoLiteral.Text = "OnCommand event: " + e.CommandName + ", ID: " +
e.CommandArgument;
        SetSelectedNode(e.CommandArgument as string);
    }

    protected void SetSelectedNode(string id)
    {
        foreach (var node in tree1.ItemsFlatList)
        {
            node.Selected = (node.Id == id);
        }
    }

    #region tree-data helpers
    private List<Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode> MakeTreeData(int
```

```
numberOfRootFolders, int numberOfChildrenPerFolder)
    {
        var data = new List<Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode>();
        for (var rootCounter = 0; rootCounter < numberOfRootFolders; rootCounter++)
        {
            var rootNode = MakeNode(rootCounter);
            data.Add(rootNode);
            for (var childCounter = 0; childCounter < numberOfChildrenPerFolder;
childCounter++)
            {
                MakeNode(childCounter, rootNode);
            }
        }
        return data;
    }

    private static TreeNode MakeNode(int index, ITreeNode parentNode = null)
    {
        var node = new TreeNode();
        node.CanHaveChildren = true;
        if (parentNode != null)
        {
            node.Id = parentNode.Id + "_Child" + index.ToString();
            parentNode.HasChildren = true;
            (parentNode.Items ?? (parentNode.Items = new
Ektron.Cms.Framework.UI.Tree.TreeNodeCollection())).Add(node);
        }
        else
        {
            node.Id = "Folder" + index.ToString();
        }
        node.Text = node.Id;
        node.CommandName = "DoCustomCommand";
        node.CommandArgument = node.Id;
        return node;
    }
    #endregion
}
```

CustomerServerTreeTemplate.ascx.cs

```
using System;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.UI;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;
using Ektron.Cms.Framework.UI.Tree;
using Ektron.Cms.Interfaces.Context;
using System.Web.UI.HtmlControls;

public partial class CustomServerTreeTemplate : TreeTemplateBase
{

    private ICmsContextService _iCmsContextService;
```

```
    public CustomServerTreeTemplate()
    {
        DefaultCssClass = "ektron-ui-control ektron-ui-tree ektron-ui-genericTree";
    }

    protected ICmsContextService CmsContextService
    {
        get { return _iCmsContextService ?? (_iCmsContextService =
ServiceFactory.CreateCmsContextService()); }
    }

    protected override PlaceHolder ControlPlaceHolder { get { return placeHolder; } }

    public override string SerializeData(Ektron.Cms.Framework.UI.Tree.TreeNodeCollection
data, bool ajaxCallback)
    {
        var childTemplate = Page.LoadControl(Request.FilePath.Substring(0,
Request.FilePath.LastIndexOf("/")) + "CustomServerTreeItemTemplate.ascx");
        childContainerPlaceHolder.Controls.Add(childTemplate);
        var childNodeRepeater = (Repeater)childTemplate.FindControl("nodeRepeater");
        childNodeRepeater.DataSource = data;
        childNodeRepeater.DataBind();

        return null;
    }

    protected override void OnPreRender(EventArgs e)
    {
        base.OnPreRender(e);

        if (this.Visible)
        {
            Packages.EktronCoreJS.Register(this);
            Packages.jQuery.Plugins.Tmpl.Register(this);
            Packages.Ektron.JSON.Register(this);
            Packages.jQuery.jQueryUI.Widget.Register(this);

            ICmsContextService cmsContext = ServiceFactory.CreateCmsContextService();
            Ektron.Cms.Framework.UI.JavaScript.Register(this, cmsContext.UIPath +
"/js/Ektron/Controls/EktronUI/Ektron.Controls.EktronUI.Tree.js");

            JavaScript.RegisterJavaScriptBlock(this.placeHolder,
base.GetInitializationScript(null), true);
        }
    }
}
```

CustomServerTreeItemTemplate.ascx.cs

```
using System;
using System.Web;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.UI;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;
using Ektron.Cms.Framework.UI.Tree;
using Ektron.Cms.Interfaces.Context;
```

```
using System.Web.UI.HtmlControls;

public partial class CustomServerTreeItemTemplate : System.Web.UI.UserControl
{

    private ICmsContextService _iCmsContextService;

    public CustomServerTreeItemTemplate() { }

    protected ICmsContextService CmsContextService
    {
        get { return _iCmsContextService ?? (_iCmsContextService =
ServiceFactory.CreateCmsContextService()); }
    }

    protected void HandleItemDataBound(Object sender, RepeaterItemEventArgs e)
    {
        if (e != null && e.Item != null && e.Item.DataItem != null)
        {
            var node = e.Item.DataItem as Ektron.Cms.Framework.UI.Tree.ITreeNode;
            if (node != null && (e.Item.ItemType == ListItemType.Item || e.Item.ItemType
== ListItemType.AlternatingItem))
            {
                e.Item.Visible = node.Visible;
                ((Literal)e.Item.FindControl("commandName")).Text = string.IsNullOrEmpty
(node.CommandName) ? null : "data-ektron-CommandName=\"" + node.CommandName + "\"";
                ((Literal)e.Item.FindControl("commandArgument")).Text =
string.IsNullOrEmpty(node.CommandArgument) ? null : "data-ektron-commandArgument=\"" +
node.CommandArgument + "\"";


                HtmlContainerControl control;
                if (string.IsNullOrEmpty(node.NavigateUrl))
                {
                    control = ((HtmlContainerControl)e.Item.FindControl("noUrlSpan"));
                    SetAttributeIfValid(control, "data-ektron-action", node.Action);
                    SetAttributeIfValid(control, "onclick", node.OnClientClick);
                    control.Attributes.Add("class", "customTextWrapper" + (node.Selected
? " selected" : ""));
                    control.InnerText = node.Text + (node.Selected ? " (selected)" :
"");
                    control.Visible = true;
                }

                if (node.Items != null && node.Items.Count > 0)
                {
                    var childContainer = ((HtmlGenericControl)e.Item.FindControl
("childContainer"));
                    childContainer.Visible = true;
                    var childTemplate = Page.LoadControl(Request.FilePath.Substring(0,
Request.FilePath.LastIndexOf("/")) + "CustomServerTreeItemTemplate.ascx");
                    childContainer.Controls.Add(childTemplate);
                    var childNodeRepeater = (Repeater)childTemplate.FindControl
("nodeRepeater");
                    childNodeRepeater.DataSource = node.Items;
                    childNodeRepeater.DataBind();
```

```
                }
            }
        }
    }

    protected void SetAttributeIfValid(HtmlControl control, string attributeName, string
attributeValue)
    {
        if (control != null && !string.IsNullOrEmpty(attributeValue))
        {
            control.Attributes.Add(attributeName, attributeValue);
        }
    }
}
```

Tree custom-template example

TreeDemo.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <b><asp:Literal ID="InfoLiteral" runat="server" Text="please click a tree node"
/></b><br /><br />
    <ektronUI:Tree ID="tree1" runat="server"
        TemplatePath="CustomTreeTemplate.ascx"
        OnCommand="OnCommand"
        />
    <style type="text/css">
        .customTextWrapper.selected { font-weight: bold;color: red; }
    </style>
    </form>
</body>
</html>
```

CustomTreeTemplate.ascx

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="CustomTreeTemplate.ascx.cs"
Inherits="CustomTreeTemplate" %>
<div id="<%= ClientID %>" <%= CssManager %> >

    <asp:PlaceHolder ID="placeHolder" runat="server" />

    <ul id="<%= ClientID %>_TreeRootElement" class="ektron-ui-tree-root" ></ul>

    <script id="<%= ClientID %>_ItemTemplate" type="text/x-jquery-tmpl">
        {{if Visible}}
```

```
            <li data-ektron-id="${Id}" class="customNode"
                {{if CommandName != null && CommandName.length > 0}} data-ektron-
CommandName="${CommandName}" data-ektron-CommandArgument="${CommandArgument}" {{/if}}
            >
                <span class="customTextWrapper {{if Selected}}selected{{/if}}" >
                    ${Text} {{if Selected}}(selected){{/if}}
                </span>
                {{if HasChildren}}
                    {{tmpl "#<%= ClientID %>_ChildrenTemplate"}}
                {{/if}}
            </li>
        {{/if}}
    </script>

    <script id="<%= ClientID %>_ChildrenTemplate" type="text/x-jquery-tmpl">
        <ul data-ektron-id="${Id}_childContainer" >
            {{if Items != null}}
                {{tmpl(Items) "#<%= ClientID %>_ItemTemplate"}}
            {{/if}}
        </ul>
    </script>
</div>
```

TreeDemo.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.UI.Tree;
using TreeNode = Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode;

public partial class TreeDemo : System.Web.UI.Page
{
    protected override void OnInit(EventArgs e)
    {
        base.OnInit(e);
        if (!Page.IsPostBack)
        {
            tree1.DataSource = MakeTreeData(5, 3);
            tree1.DataBind();
        }
    }

    protected void OnCommand(object sender, CommandEventArgs e)
    {
        InfoLiteral.Text = "OnCommand event: " + e.CommandName + ", ID: " +
e.CommandArgument;
        SetSelectedNode(e.CommandArgument as string);
    }

    protected void SetSelectedNode(string id)
    {
        foreach (var node in tree1.ItemsFlatList)
        {
            node.Selected = (node.Id == id);
```

```
        }
    }

    #region tree-data helpers
    private List<Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode> MakeTreeData(int
numberOfRootFolders, int numberOfChildrenPerFolder)
    {
        var data = new List<Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode>();
        for (var rootCounter = 0; rootCounter < numberOfRootFolders; rootCounter++)
        {
            var rootNode = MakeNode(rootCounter);
            data.Add(rootNode);
            for (var childCounter = 0; childCounter < numberOfChildrenPerFolder;
childCounter++)
            {
                MakeNode(childCounter, rootNode);
            }
        }
        return data;
    }

    private static TreeNode MakeNode(int index, ITreeNode parentNode = null)
    {
        var node = new TreeNode();
        node.CanHaveChildren = true;
        if (parentNode != null)
        {
            node.Id = parentNode.Id + "_Child" + index.ToString();
            parentNode.HasChildren = true;
            (parentNode.Items ?? (parentNode.Items = new
Ektron.Cms.Framework.UI.Tree.TreeNodeCollection())).Add(node);
        }
        else
        {
            node.Id = "Folder" + index.ToString();
        }
        node.Text = node.Id;
        node.CommandName = "DoCustomCommand";
        node.CommandArgument = node.Id;
        return node;
    }
    #endregion
}
```

CustomTreeTemplate.ascx.cs

```
using System;
using System.Web;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.UI;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;
using Ektron.Cms.Framework.UI.Tree;
using Ektron.Cms.Interfaces.Context;

public partial class CustomTreeTemplate : TreeTemplateBase {
```

```
    public CustomTreeTemplate() {
        DefaultCssClass = "ektron-ui-control ektron-ui-tree ektron-ui-genericTree";
    }

    protected override PlaceHolder ControlPlaceHolder { get { return placeHolder; } }

    public override string SerializeData(Ektron.Cms.Framework.UI.Tree.TreeNodeCollection
data, bool ajaxCallback) {
        return JSON = base.SerializeData(data, ajaxCallback);
    }

    protected override void OnPreRender(EventArgs e) {
        base.OnPreRender(e);

        if (this.Visible) {
            Packages.EktronCoreJS.Register(this);
            Packages.jQuery.Plugins.Tmpl.Register(this);
            Packages.Ektron.JSON.Register(this);
            Packages.jQuery.jQueryUI.Widget.Register(this);

            ICmsContextService cmsContext = ServiceFactory.CreateCmsContextService();
            Ektron.Cms.Framework.UI.JavaScript.Register(this, cmsContext.UIPath +
"/js/Ektron/Controls/EktronUI/Ektron.Controls.EktronUI.Tree.js");

            JavaScript.RegisterJavaScriptBlock(this.placeHolder,
base.GetInitializationScript(JSON), true);
        }
    }
}
```

Tree in repeater example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <asp:Repeater ID="repeater1" runat="server" >
        <HeaderTemplate>
            <table >
        </HeaderTemplate>
        <ItemTemplate>
            <tr>
                <td><%# Eval("Year") %></td>
                <td>*<%# Eval("Make") %>*</td>
                <td>
                    <ektronUI:Tree ID="tree1" runat="server" OnItemDataBound="Tree_
```

```
ItemDataBound" DataSource='<%# Container.DataItem %>' SelectionMode="SingleForAll" >
                        <ektronUI:TreeItem Id="Base" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "Model") %>' OnClientClick="alert('A salesman will
contact you on this vehicle!');" />
                        <ektronUI:TreeItem Id="Sport" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "Model") %>'  OnClientClick="alert('A salesman will
contact you on this vehicle!');"/>
                        <ektronUI:TreeItem Id="Touring" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "Model") %>'  OnClientClick="alert('A salesman will
contact you on this vehicle!');">
                            <ektronUI:TreeItem Id="Convertible" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "Model") %>' OnClientClick="alert('A salesman will
contact you on this vehicle!');" />
                        </ektronUI:TreeItem>
                    </ektronUI:Tree>
                </td>
            </tr>
        </ItemTemplate>
        <FooterTemplate>
            </table>
        </FooterTemplate>
    </asp:Repeater>
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;

public partial class TreeDemo : System.Web.UI.Page
{
    public class Car
    {
        public string Make { get; set; }
        public string Model { get; set; }
        public string Color { get; set; }
        public string Year { get; set; }
    }

    protected override void OnInit(EventArgs e)
    {
        base.OnInit(e);

        var list = new List<Car>() {
            new Car() { Make="BMW", Model="Z4", Color="Red", Year="2013" },
            new Car() { Make="Ford", Model="Mustang", Color="Yellow", Year="2012" },
            new Car() { Make="Chevrolet", Model="Camaro", Color="Blue", Year="2011" }
        };
        repeater1.DataSource = list;
        repeater1.DataBind();
    }
```

```
    public void Tree_ItemDataBound(Object sender,
TreeItemDataBoundEventArgs<Ektron.Cms.Framework.UI.Controls.EktronUI.TreeNode> e)
    {
        var treeNode = e.TreeNode;
        var id = treeNode.Id;
        treeNode.Text += " - Package: " + id;
    }
}
```

Tree multi-select example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <ektronUI:Tree ID="tree1" runat="server" ShowRoot="true" SelectionMode="MultiForAll"
>
        <ektronUI:TreeItem Id="item1" runat="server" Text="Item-One" Selected="True" />
        <ektronUI:TreeItem Id="item2" runat="server" Text="Item-Two" Selected="True" />
        <ektronUI:TreeItem Id="item3" runat="server" Text="Item-Three" Selected="True"
/>
    </ektronUI:Tree>
    </form>
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

Tree single-select example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
```

```
</head>
<body>
    <form id="form1" runat="server">
    <ektronUI:Tree ID="tree1" runat="server" ShowRoot="true"
SelectionMode="SingleForAll" >
        <ektronUI:TreeItem Id="item1" runat="server" Text="Item-One" Selected="True" />
        <ektronUI:TreeItem Id="item2" runat="server" Text="Item-Two" />
        <ektronUI:TreeItem Id="item3" runat="server" Text="Item-Three" />
    </ektronUI:Tree>
    </form>
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

ContextMenu Tree

9.10 and higher

Creates a context menu, populated by declarative data.

ContextMenuTree example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <ektronUI:ContextMenuTree ID="ContextMenuTree1" runat="server" >
        <ektronUI:ContextMenuTreeItem Id="item1" runat="server" Text="Item One"
OnClientClick="alert('Item One Clicked');" />
        <ektronUI:ContextMenuTreeItem Id="item2" runat="server" Text="Item Two"
OnClientClick="alert('Item Two Clicked');" />
        <ektronUI:ContextMenuTreeItem Id="item3" runat="server" Text="Item Three"
OnClientClick="alert('Item Three Clicked');" >
            <ektronUI:ContextMenuTreeItem Id="item3_a" runat="server" Text="Item Three-
A" OnClientClick="alert('Item Three-A Clicked');" />
            <ektronUI:ContextMenuTreeItem Id="item3_b" runat="server" Text="Item Three-
B" OnClientClick="alert('Item Three-B Clicked');" />
        </ektronUI:ContextMenuTreeItem>
    </ektronUI:ContextMenuTree>
    </form>
```

```
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

FolderTree

- **FolderTree**. Adds a control that exposes the CMS folder structure to a page.

  NOTE: The RootId attribute is the starting location for the control, using "0" to begin at the actual CMS root folder; it can be any child folder (for example, set it to the numeric ID of a child folder taken from the workareas folder properties, and the tree will begin there).

- **FolderTree with Ajax**. 9.10 and higher Adds a Folder-Tree control that uses Ajax to fetch the data as needed, reducing the demand placed on the server when a page is first loaded. When the folder tree is very large, the control can defer fetching the data until it is needed. By setting `PageDepth="1"` the tree will get only 1 level below the current level, using an Ajax callback to get the data when a folder that contains children is first opened. By default, the `PageDepth="0"`, which sets no depth-limit and results in the entire tree data being fetched at one time.
- **FolderTree-AjaxCallbackOverride**. Alters the data returned to the tree in code-behind using the tree-controls OnCallback handler.
- **FolderTree Custom-Processor**. 9.10 and higher Adds a custom processor to the control, so that you can manipulate the data.
- **FolderTree-DataDump**. By specifying `DefaultTemplate="DataDump"`, the control will output the data for viewing to aid designers understanding what properties and values are available on each node, to bind with in a template (for example, in a custom template).
- **FolderTree-ItemDataBound**. Adds an OnItemDataBound handler in code-behind that modifies each node's text.
- **FolderTree-Linq**. Expands to, and selects, a target node using LINQ.

  NOTE: This demo assumes that the OnTrek sample site is installed, otherwise the code will have to be tweaked to find folder nodes in your site.

- **FolderTree-PreSerialize**. Adds an OnPreSerializeData handler in code-behind that modifies each nodes text .
- **FolderTree Selections**. Determines which nodes have been selected in code-behind.

- **FolderTree Selections (auto-postback)**. Determines which nodes have been selected in code-behind, also using the tree controls auto-postback feature to automatically update the page when a node is selected.
- **FolderTree-ServerRender**. By specifying `DefaultTemplate="ServerRender"`, the control uses server-side templates to bind with to generate the rendered markup. The server side template uses ASP.NET server controls. You can copy the template to a custom version to alter the markup. If no `DefaultTemplate` is specified, or if it is set to `ClientRender`, then the data is sent to the client side along with templates for binding within the browser using JavaScript and the jQuery Templating Plugin.

  NOTE: While server-side rendering may simplify the creation of custom templates for ASP.NET webform developers, it places additional burdens on the server that may not be the best option for performance reasons.

- **Paged FolderTree**. By specifying the `PageSize` attribute, the control limits the number of folders fetched per parent folder to the given value, and show a link that the user can click to see more folders. You can alter the text of the link using the `PagingText` attribute.

FolderTree example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <ektronUI:FolderTree ID="folderTree1" runat="server" RootId="0"
UseInternalAdmin="true" />
    </form>
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

FolderTree with Ajax example

9.10 and higher

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <!--
        When the folder tree is very large, the control can defer fetching the data
until it is needed:
        By setting PageDepth="1" the tree will only get one level below the current
level (e.g. the root
        level) at a time, utilizing an Ajax callback to get the data when a folder that
contains children
        is first opened.
        By default, the PageDepth="0", which sets no depth-limit and results in the
entire tree data
        being fetched at one time.
    -->
    <ektronUI:FolderTree ID="folderTree1" runat="server"
        RootId="0"
        PageDepth="1"
        UseInternalAdmin="true"
        />
    </form>
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

FolderTree-AjaxCallbackOverride example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
```

```
    <form id="form1" runat="server">
    <!--
        By setting PageDepth="1" the tree will only get one level below
        the current level (e.g. the root level) at a time, forcing an
        Ajax callback that we can use
    -->
    <ektronUI:FolderTree ID="folderTree1" runat="server"
        RootId="0"
        PageSize="0" PageDepth="1"
        SelectionMode="MultiForAll"
        UseInternalAdmin="true"
        OnCallback="HandleCallback"
        BeforeAjax="AddCustomPayload"
        />
        <script type="text/javascript">
            function AddCustomPayload(obj) {
                // add a custom payload, to be used on the
                // server-side, in the Ajax "OnCallback" handler"
                obj.data = obj.data + "#customPayload=xyz";
            }
        </script>
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;

public partial class TreeDemo : System.Web.UI.Page
{
    protected void HandleCallback(object source, TreeAjaxCallbackEventArgs e)
    {
        if (e.EventArgument.Contains("#customPayload=xyz"))
        {
            e.AbortDefaultProcessing = true;
            folderTree1.Items.Clear();

            var node = new FolderTreeNode();
            node.Id = "ektronHome";
            node.Text = "Ektron";
            node.NavigateUrl = "http://www.ektron.com";
            folderTree1.Items.Add(node);

            node = new FolderTreeNode();
            node.Id = "msnHome";
            node.Text = "MSN";
            node.NavigateUrl = "http://www.msn.com";
            folderTree1.Items.Add(node);
```

```
        }
    }
}
```

FolderTree-CustomProcessor example

9.10 and higher

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
        <ektronUI:FolderTree ID="folderTree1" runat="server"
            RootId="0"
            UseInternalAdmin="true">
        </ektronUI:FolderTree>
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;

public partial class TreeDemo : System.Web.UI.Page, ITreeDataProcessor
{
    protected override void OnLoad(EventArgs e)
    {
        base.OnLoad(e);
        folderTree1.AddProcessor(this);
    }

    #region ITreeDataProcessor Members
    public ITreeDataProcessor InnerProcessor { get; set; }

    public Ektron.Cms.Framework.UI.Tree.TreeNodeCollection Process
(Ektron.Cms.Framework.UI.Tree.TreeNodeCollection data, TreeDataContext context, Object
flatData = null)
    {
        var modData = data;
```

```
        if (InnerProcessor != null)
        {
            modData = InnerProcessor.Process(modData, context, flatData);
        }

        return ModifyData(modData);
    }
    #endregion

    protected Ektron.Cms.Framework.UI.Tree.TreeNodeCollection ModifyData
(Ektron.Cms.Framework.UI.Tree.TreeNodeCollection data)
    {
        foreach (var item in data)
        {
            if (item.CanHaveChildren)
            {
                item.Text = item.Text + " (in Custom-Processor)";
            }

            if (item.Items != null && item.Items.Count > 0)
            {
                ModifyData(item.Items);
            }
        }

        return data;
    }
}
```

FolderTree-DataDump example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <!--
        By specifying DefaultTemplate="DataDump", the control will output the data
        for viewing to aid designers understand what properties and values are
        available for use to bind with in a template (e.g. a custom template).
    -->
    <ektronUI:FolderTree ID="folderTree1" runat="server"
        RootId="0"
        UseInternalAdmin="true"
        DefaultTemplate="DataDump" />
    </form>
```

```
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

FolderTree-ItemDataBound example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <ektronUI:FolderTree ID="folderTree1" runat="server"
        RootId="0"
        OnItemDataBound="OnItemDataBound"
        UseInternalAdmin="true"
        />
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;

public partial class TreeDemo : System.Web.UI.Page
{
    protected override void OnLoad(EventArgs e)
    {
        base.OnLoad(e);
        folderTree1.DataBind();
    }

    public void OnItemDataBound(Object sender,
TreeItemDataBoundEventArgs<FolderTreeNode> e)
```

```
    {
        e.TreeNode.Text += " (modified in OnItemDataBound)";
    }
}
```

FolderTree-Linq example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <ektronUI:FolderTree ID="folderTree1" runat="server"
        RootId="0"
        OnPreSerializeData="PreSerializeHandler"
        UseInternalAdmin="true"
        />
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Ektron.Cms.Framework.UI.Tree;

public partial class TreeDemo : System.Web.UI.Page
{
    protected void PreSerializeHandler(object sender, EventArgs e) {
        var flatList = folderTree1.ItemsFlatList;

        // use Linq to simplify selecting target node(s):

        var selectedNodes =
            from selectedNode in flatList
            // Note: Any of the following will select the "General Questions" child
node:
            where selectedNode.Id == "122"
            //where selectedNode.Text == "General Questions"
            //where selectedNode.Path == "0^121^71^77^90^122"
            //where selectedNode.Level == 5 // note, this selects all nodes at level 5
            select selectedNode;

        foreach (var selectedItem in selectedNodes) {
            selectedItem.Selected = true;
```

```
            if (selectedItem.CanHaveChildren) {
                selectedItem.Expanded = true;
            }

            ExpandAncestors(selectedItem, flatList);
        }

        var filteredNodes =
            from item in flatList
            where item.Level == 0
            select item;

        folderTree1.Items.Clear();
        folderTree1.Items.AddRange(filteredNodes);
    }

    public void ExpandAncestors<T>(T node, IList<T> flatList, bool select = false)
        where T : class, ITreeNode, new() {
        var parentNode = GetParent<T>(node, flatList);
        if (parentNode != null) {
            parentNode.Expanded = true;
            parentNode.Selected = parentNode.Selected || select;
            ExpandAncestors<T>(parentNode, flatList);
        }
    }

    public T GetParent<T>(T node, IList<T> flatList)
        where T : class, ITreeNode, new() {
        if (node != null && !string.IsNullOrEmpty(node.ParentId) && node.ParentId !=
node.Id && flatList != null) {
            var parentId = node.ParentId;
            var parentNode = (
                from f in flatList
                where f.Id == parentId
                select f
                ).First();
            return parentNode;
        }
        return null;
    }
}
```

FolderTree Selections example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
```

```
<body>
    <form id="form1" runat="server">
    <asp:ScriptManager runat="server" ID="ScriptManager1" />
    <asp:UpdatePanel ID="up1" runat="server" ChildrenAsTriggers="True" >
        <ContentTemplate>
            <ektronUI:FolderTree ID="folderTree1" runat="server"
                    RootId="0" SelectionMode="MultiForAll"
                    UseInternalAdmin="true"
                    OnSelectionEvent="OnSelectionEvent"
                    OnClick="OnClick"
                    />
            <br />
            <input type="submit" name="submitButton" value="Submit Selections" /><br />
            <h3>Selected Item IDs:</h3>
            <asp:TextBox ID="tb1" runat="server" TextMode="MultiLine" Rows="10" />
        </ContentTemplate>
    </asp:UpdatePanel>
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;

public partial class TreeDemo : System.Web.UI.Page
{
    protected override void OnLoad(EventArgs e)
    {
        base.OnLoad(e);
        tb1.Text = "";
    }

    public void OnSelectionEvent(Object sender, TreeSelectionEventArgs argument)
    {
        if (argument.Ids.Length > 0)
        {
            string result = "";
            foreach (var id in argument.Ids)
            {
                result += id + " selected" + Environment.NewLine;
            }
            tb1.Text = result;
        }
    }

    public void OnClick(object sender, EventArgs e) { }
}
```

FolderTree Selections (auto-postback) example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <asp:ScriptManager runat="server" ID="ScriptManager1" />
    <asp:UpdatePanel ID="up1" runat="server" ChildrenAsTriggers="True" >
        <ContentTemplate>
            <ektronUI:FolderTree ID="folderTree1" runat="server"
                    RootId="0" SelectionMode="MultiForAll"
                    UseInternalAdmin="true"
                    OnSelectionEvent="OnSelectionEvent"
                    OnClick="OnClick"
                    AutoPostBackOnSelectionChanged="True"
                    />
            <br />
            <h3>Selected Item IDs:</h3>
            <asp:TextBox ID="tb1" runat="server" TextMode="MultiLine" Rows="10" />
        </ContentTemplate>
    </asp:UpdatePanel>
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;

public partial class TreeDemo : System.Web.UI.Page
{
    protected override void OnLoad(EventArgs e)
    {
        base.OnLoad(e);
        tb1.Text = "";
    }

    public void OnSelectionEvent(Object sender, TreeSelectionEventArgs argument)
    {
        if (argument.Ids.Length > 0)
        {
            string result = "";
            foreach (var id in argument.Ids)
            {
                result += id + " selected" + Environment.NewLine;
```

```
            }
            tbl.Text = result;
        }
    }

    public void OnClick(object sender, EventArgs e) { }
}
```

FolderTree-PreSerialize example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <ektronUI:FolderTree ID="folderTree1" runat="server"
        RootId="0"
        UseInternalAdmin="true"
        OnPreSerializeData="PreSerializeHandler"
        />
    </form>
</body>
</html>
```

.aspx.cs

```
using System;

public partial class TreeDemo : System.Web.UI.Page
{
    protected void PreSerializeHandler(object sender, EventArgs e) {
        if (folderTree1 != null && folderTree1.Items is
Ektron.Cms.Framework.UI.Tree.TreeNodeCollection) {
            ModifyData(folderTree1.Items as
Ektron.Cms.Framework.UI.Tree.TreeNodeCollection);
        }
    }

    protected void ModifyData(Ektron.Cms.Framework.UI.Tree.TreeNodeCollection data) {
        foreach (var item in data) {
            if (item.CanHaveChildren) {
                item.Text += " (modified data during OnPreSerializeData event)";
            }
            if (item.Items != null && item.Items.Count > 0) {
                ModifyData(item.Items);
            }
        }
```

```
    }
}
```

FolderTree-ServerRender example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <!--
        By specifying DefaultTemplate="ServerRender", the control will utilize
        server-side templates to bind with to generate the rendered markup.
        The server side template use ASP.NET server controls.
        The template can be copied to a custom version to alter the markup.
        If no DefaultTemplate is specified, or if it is set to ClientRender,
        then the data is sent to the client side along with templates for
        binding on the within the browser using JavaScript and the Temple
        templating engine.
    -->
    <ektronUI:FolderTree ID="folderTree1" runat="server"
        RootId="0"
        UseInternalAdmin="true"
        DefaultTemplate="ServerRender"
        />
    </form>
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

Paged FolderTree example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <!--
        By specifying the PageSize attribute, the control will limit
        the number of folders fetched per parent folder to the given value,
        and show a link that the user can click to see more folders.
        The text of the link can be altered using the PagingText attribute.
    -->
    <ektronUI:FolderTree ID="folderTree1" runat="server"
        RootId="0"
        PageSize="5"
        PageDepth="1"
        UseInternalAdmin="true"
        />
    </form>
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

MenuTree examples

- **MenuTree**. Adds a control that exposes a CMS Menu structure to the page.

  NOTE: The RootId attribute must be set to the desired CMS menu ID.

- **MenuTree-DataDump**. By specifying `DefaultTemplate="DataDump"`, the control will output the data for viewing to aid designers understanding what properties and values are available on each node, to bind with in a template (for example, in a custom template).
- **MenuTree-ServerRende**r. By specifying `DefaultTemplate="ServerRender"`, the control will utilize server-side templates to bind with to generate the rendered markup. The server side template use ASP.NET server controls. The template can be copied to a custom version to alter the markup. If no DefaultTemplate is specified, or if it is set to ClientRender, then the data is sent to the client side along with templates for binding within the browser using JavaScript and the jQuery Templating Plugin.

  NOTE: While server-side rendering may simplify the creation of custom templates for ASP.NET webform developers, it places additional burdens on the server that may not be the best option for performance reasons.

MenuTree example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    (Note that the RootId attribute must be set to the desired CMS menu ID)<br /><br />
    <style type="text/css">
        .demoTree.ektron-ui-tree.ektron-ui-menuTree { font-size: 12px; }
        .demoTree.ektron-ui-tree.ektron-ui-menuTree ul.ektron-ui-tree-root .ektron-ui-
tree-menu { width: 115px;}
        .demoTree.ektron-ui-tree.ektron-ui-menuTree li ul, .ektron-ui-tree.ektron-ui-
menuTree ul li ul li ul { width: 115px;}
        .demoTree.ektron-ui-tree.ektron-ui-menuTree ul.ektron-ui-tree-root li ul { top:
29px;}
    </style>
    <ektronUI:MenuTree ID="MenuTree1" runat="server"
        RootId="6"
        CssClass="demoTree"
        ShowRoot="false"
        UseInternalAdmin="true"
        >
        <ektronUI:MenuTreeItem Id="declarativeLink" runat="server"
            Text="Ektron"
            NavigateUrl="http://www.ektron.com"
            />
    </ektronUI:MenuTree>
    </form>
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

MenuTree-DataDump example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    <style type="text/css">
        .demoTree.ektron-ui-tree.ektron-ui-menuTree { font-size: 12px; }
        .demoTree.ektron-ui-tree.ektron-ui-menuTree ul.ektron-ui-tree-root .ektron-ui-
tree-menu { width: 115px;}
        .demoTree.ektron-ui-tree.ektron-ui-menuTree li ul, .ektron-ui-tree.ektron-ui-
menuTree ul li ul li ul { width: 115px;}
        .demoTree.ektron-ui-tree.ektron-ui-menuTree ul.ektron-ui-tree-root li ul { top:
29px;}
    </style>
    <!--
        By specifying DefaultTemplate="DataDump", the control will output the data
        for viewing to aid designers understand what properties and values are
        available for use to bind with in a template (e.g. a custom template).
    -->
    <ektronUI:MenuTree ID="MenuTree1" runat="server" RootId="6" CssClass="demoTree"
        ShowRoot="false"
        UseInternalAdmin="true"
        DefaultTemplate="DataDump"
        >
        <ektronUI:MenuTreeItem Id="declarativeLink" runat="server"
            Text="Ektron"
            NavigateUrl="http://www.ektron.com"
            />
    </ektronUI:MenuTree>
    </form>
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

MenuTree-ServerRender example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
```

```
<body>
    <form id="form1" runat="server">
    <style type="text/css">
        .demoTree.ektron-ui-tree.ektron-ui-menuTree { font-size: 12px; }
        .demoTree.ektron-ui-tree.ektron-ui-menuTree ul.ektron-ui-tree-root .ektron-ui-
tree-menu { width: 115px;}
        .demoTree.ektron-ui-tree.ektron-ui-menuTree li ul, .ektron-ui-tree.ektron-ui-
menuTree ul li ul li ul { width: 115px;}
        .demoTree.ektron-ui-tree.ektron-ui-menuTree ul.ektron-ui-tree-root li ul { top:
29px;}
    </style>
    <!--
        By specifying DefaultTemplate="ServerRender", the control will utilize
        server-side templates to bind with to generate the rendered markup.
        The server side template uses ASP.NET server controls.
        The template can be copied to a custom version to alter the markup.
        If no DefaultTemplate is specified, or if it is set to ClientRender,
        then the data is sent to the client side along with templates for
        binding on the within the browser using JavaScript and the Temple
        templating engine.
    -->
    <ektronUI:MenuTree ID="MenuTree1" runat="server" RootId="6" CssClass="demoTree"
        ShowRoot="false" UseInternalAdmin="true" >
        <ektronUI:MenuTreeItem Id="declarativeLink" runat="server" Text="Ektron"
NavigateUrl="http://www.ektron.com" />
    </ektronUI:MenuTree>
    </form>
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

TaxonomyTree examples

- **TaxonomyTree**. Adds a control that exposes a CMS taxonomy structure to the page.
- **TaxonomyTree (auto-postback)**. Determines which taxonomy nodes have been selected in code-behind, also using the tree controls auto-postback feature, to automatically update the page when a node is selected.
- **TaxonomyTree-DataDump**. By specifying `DefaultTemplate="DataDump"`, the control will output the data for viewing to aid designers understanding what properties and values are available on each node, to bind with in a template (for example, in a custom template).
- **TaxonomyTree-ServerRender**. By specifying `DefaultTemplate="ServerRender"`, the control uses server-side templates to bind with to generate the rendered markup. The server side template uses ASP.NET server controls. You can copy the template to a custom version to alter

the markup. If no `DefaultTemplate` is specified, or if it is set to `ClientRender`, then the data is sent to the client side along with templates for binding within the browser using JavaScript and the jQuery Templating Plugin.

NOTE: While server-side rendering may simplify the creation of custom templates for ASP.NET webform developers, it places additional burdens on the server that may not be the best option for performance reasons.

TaxonomyTree example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    (Note that the RootId attribute must be set to the desired CMS taxonomy ID)<br /><br
/>
    <ektronui:TaxonomyTree ID="tree1" runat="server"
                RootId="189" ShowRoot="false" UseInternalAdmin="true"
                ShowItemCounts="true" ShowTaxonomyItems="true"
                SelectionMode="MultiForAll"
                OnSelectionEvent="HandleOnSelectionEvent"
                />
    <br />
    <input type="submit" name="submitButton" value="Submit Selections" /><br />
    <h3>Selected Item IDs:</h3>
    <asp:TextBox ID="tb1" runat="server" TextMode="MultiLine" Rows="10" />
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;

public partial class TreeDemo : System.Web.UI.Page
{
    public void HandleOnSelectionEvent(object sender, TreeSelectionEventArgs argument)
    {
        if (argument.Ids.Length > 0)
        {
            string result = "";
            foreach (var id in argument.Ids)
            {
                result += id + " selected" + Environment.NewLine;
            }
```

```
            tb1.Text = result;
        }
    }
}
```

TaxonomyTree (auto-postback) example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    (Note that the RootId attribute must be set to the desired CMS taxonomy ID)<br /><br
/>
    <ektronui:TaxonomyTree ID="tree1" runat="server"
                RootId="189" ShowRoot="false" UseInternalAdmin="true"
                ShowItemCounts="true" ShowTaxonomyItems="true"
                SelectionMode="MultiForAll"
                OnSelectionEvent="HandleOnSelectionEvent"
                AutoPostBackOnSelectionChanged="True"
                />
    <br />
    <h3>Selected Item IDs:</h3>
    <asp:TextBox ID="tb1" runat="server" TextMode="MultiLine" Rows="10" />
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;

public partial class TreeDemo : System.Web.UI.Page
{
    public void HandleOnSelectionEvent(object sender, TreeSelectionEventArgs argument)
    {
        if (argument.Ids.Length > 0)
        {
            string result = "";
            foreach (var id in argument.Ids)
            {
                result += id + " selected" + Environment.NewLine;
            }
            tb1.Text = result;
        }
```

```
    }
}
```

TaxonomyTree-DataDump example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    (Note that the RootId attribute must be set to the desired CMS taxonomy ID)<br /><br
/>
    <!--
        By specifying DefaultTemplate="DataDump", the control will output the data
        for viewing to aid designers understand what properties and values are
        available for use to bind with in a template (e.g. a custom template).
    -->

    <ektronui:TaxonomyTree ID="tree1" runat="server"
        RootId="189"
        ShowRoot="false"
        ShowItemCounts="true"
        ShowTaxonomyItems="true"
        DefaultTemplate="DataDump"
        UseInternalAdmin="true"
        />
    <br />
    <input type="submit" name="submitButton" value="Submit Selections" /><br />
    <h3>Selected Item IDs:</h3>
    <asp:TextBox ID="tb1" runat="server" TextMode="MultiLine" Rows="10" />
    </form>
</body>
</html>
```

.aspx.cs

```
public partial class TreeDemo : System.Web.UI.Page
{

}
```

TaxonomyTree-SeverRender example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="TreeDemo.aspx.cs"
Inherits="TreeDemo" ValidateRequest="false" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>TreeDemo</title>
</head>
<body>
    <form id="form1" runat="server">
    (Note that the RootId attribute must be set to the desired CMS taxonomy ID)<br /><br
/>
    <!--
        By specifying DefaultTemplate="ServerRender", the control will utilize
        server-side templates to bind with to generate the rendered markup.
        The server side template uses ASP.NET server controls.
        The template can be copied to a custom version to alter the markup.
        If no DefaultTemplate is specified, or if it is set to ClientRender,
        then the data is sent to the client side along with templates for
        binding on the within the browser using JavaScript and the Temple
        templating engine.
    -->
    <ektronui:TaxonomyTree ID="tree1" runat="server"
        RootId="189" ShowRoot="false"
        UseInternalAdmin="true"
        ShowItemCounts="true"
        ShowTaxonomyItems="true"
        SelectionMode="MultiForAll"
        OnSelectionEvent="HandleOnSelectionEvent"
        DefaultTemplate="ServerRender"
        />
    <br />
    <input type="submit" name="submitButton" value="Submit Selections" /><br />
    <h3>Selected Item IDs:</h3>
    <asp:TextBox ID="tb1" runat="server" TextMode="MultiLine" Rows="10" />
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Widgets;

public partial class TreeDemo : System.Web.UI.Page
{
    public void HandleOnSelectionEvent(object sender, TreeSelectionEventArgs argument)
    {
        if (argument.Ids.Length > 0)
        {
```

```
            string result = "";
            foreach (var id in argument.Ids)
            {
                result += id + " selected" + Environment.NewLine;
            }
            tb1.Text = result;
        }
    }
}
```

ValidationMessage

8.50 and higher

```
<EktronUI:ValidationMessage>
```

Displays the messages that are set in the associated server control's ErrorMessage property.

- Validation rule error messages that are defined for the Control set in the `AssociatedControlID` property are displayed as text for the `<EktronUI:ValidationMessage>` control when error events are triggered.
- This control correctly sets the for attribute of the label element to the form field embedded within an EktronUI control. An asp:Label does not set the for attribute correctly when the `AssociatedControlID` is set on an EktronUI control.
- The markup is always an HTML `<label>` tag. An asp:Label outputs either a `<span>` or a `<label>` tag depending on whether the `AssociatedControlID` property is set or not.

Properties

- **DefaultTemplate**. Gets the name of the Default Template used for rendering the `<EktronUI:TextField>` control.
- **Value**. Strongly typed value accessor.

Events for ValidationMessage

None.

Methods for ValidationMessage

None.

Theming for ValidationMessage

None.

Example for ValidationMessage

Default functionality example

.aspx

```
<ektronUI:Label id="uxTextFieldLabel" AssociatedControlID="uxTextField"
runat="server">Ektron Text Field:</ektronUI:Label>
<ektronUI:TextField ID="uxTextField" runat="server" Text="">
    <ValidationRules>
```

```
        <ektronUI:RequiredRule ErrorMessage="Text is required!" />
    </ValidationRules>
</ektronUI:TextField>
<ektronUI:ValidationMessage ID="validationMessage" runat="server"
AssociatedControlID="uxTextField" /><br />
<ektronUI:Button ID="uxButton" runat="server" Text="Submit" OnClick="uxButton_click"
></ektronUI:Button>
```

.aspx.cs

```
using System;
using Ektron.Site.Developer;

public partial class developer_Framework_UI_Controls_EktronUI_ValidationMessage_
ValidationMessage : SampleControlPage
{
    protected void uxButton_click(Object sender, EventArgs e)
    {

    }
}
```

Wizard

9.00 and higher

```
<EktronUI:Wizard>
```

Simplifies the process of creating multi-step, branching user interfaces. It lets a developer use their Web Forms skills to focus on gathering user input and processing it once, and depend on the wizard to manage the associated page events and form state.

Visually, a wizard is comprised of a list of all available steps, the form for the current step, and a set of buttons to navigate through the process (e.g. next, previous).

A developer implements a wizard by defining a series of steps using EktronUI Step and Summary controls. Any wizard only requires a single step, but a developer may choose to include a summary, define custom property values or assign handlers for its events.

Properties

- `HideStepListing`. Gets or sets a value indicating whether to hide or show the list of names and links to the steps of the wizard. Without this, the wizard does not provide the user with context for the step they are executing.
- `IncludeSummaryInStepListing`. Gets or sets a value indicating whether to include the post-finish summary step in the list of all steps. A user can never skip directly to this step.
- `OnCancelClientClick`. Gets or sets the on client-side code to execute when the user clicks the cancel button. If this code returns false, it will stop the execution of the button click. This is analogous to the `OnClientClick` property of the ASP.Net button.
- `OnCloseClientClick`. Gets or sets the on client-side code to execute when the user clicks the close button. If this code returns false, it will stop the execution of the button click. This is analogous to the `OnClientClick` property of the ASP.Net button.
- `Separator`. Gets or sets the character or string that separates a step's status from its name in the step list.
- `ToolTip`. Gets or sets the text displayed when the mouse pointer hovers over the Web server control.This is passed to the base control, but doesn't have any particular meaning for the Wizard.

Events for Wizard

- `Cancel`. This occurs when the user requests to cancel the wizard. By default, this has no implementation and no Cancel button is shown.

- `Finish`. This occurs after the final non-summary step has been completed, thus completing the wizard. This is the appropriate time to finally process the information entered into the wizard.
- `Start`. This occurs only once when the wizard is loaded onto the page the first time. This provides an opportunity prepare the first step before it is activated as well as any bootstrapping that may be necessary.

Events for Wizard Step

- `CurrentRender`. This occurs when the step is rendered to the page and is useful for operations which only need to occur when the step is displayed (for example, data-binding). This event must be able to repeat without corrupting data.
- `Finish`. This occurs when the user attempts to advance past the step and is when information from the step may be validated. This event must be able to repeat without corrupting data.
- `Start`. This occurs when the step is loaded, whether to be displayed or validated when attempting to advance to a later step. This provides an opportunity to prepare the step for validation (e.g. reflecting changes from earlier steps, changing field visibility, etc). This event must be able to repeat without corrupting data and cannot assume that a step that starts will necessarily remain current.

Methods for Wizard

- `Complete`. Executes the same completion process as would clicking the "Finish" button. If the user is not on the last step, it will advance through every step before executing the Finish event.
- `Exit`. Executes the Cancel event.
- `Go`. Advances or returns the user to the specified step, stopping if necessary due to validation failures.
- `Next`. Advances the user to the next step if the current step is valid and is not the last non-summary step.
- `Previous`. Returns the user to the preceding step if the current step is not the first.

Methods for Wizard Step

- `Activate`. Activates the step, executing the Start event.
- `Complete`. Triggers the step’s completion process: if the form passes validation and the Finish event does not cancel the process, it returns true and allows the Wizard to advance; otherwise it returns false.

Theming for Wizard

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool

to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

NOTE: The Ektron UI Wizard Step only acts as a container for its child controls and does not have any independent markup to theme.

```
<div class="steps">
<ol class="ektron-ui-clearfix">
<li class="completed enabledui-widget ui-state-default">
<a>
<span class="label">Step 1</span>
<span class="separator">:</span>
<span class="status">Choice</span>
</a>
</li>
<li class="current enabledui-widget ui-state-default ui-state-active">
<a><span class="label">Step 2</span></a>
</li>
<li class="pendingui-widget ui-state-default">
<a><span class="label">Step 3</span></a>
</li>
<li class="pending summaryui-widget ui-state-default">
<a><span class="label">Summary</span></a>
</li>
</ol>
</div>
<fieldset class="form">
<legend>Step 2</legend>
<div>…</div>
</fieldset>
<div class="navigation">
<span class="ektron-ui-control ektron-ui-button cancel">
<button aria-disabled="false" role="button" class="ui-button ui-widget ui-state-default
ui-corner-all ui-button-text-icon-primary">
<span class="ui-button-icon-primary ui-icon ui-icon-cancel"></span>
<span class="ui-button-text">Cancel</span>
</button>
</span>
<span class="ektron-ui-control ektron-ui-buttonprevious">
<button aria-disabled="false" role="button" class="ui-button ui-widget ui-state-default
ui-corner-all ui-button-text-icon-primary">
<span class="ui-button-icon-primary ui-icon ui-icon-seek-prev"></span>
<span class="ui-button-text">Previous</span>
</button>
</span>
<span class="ektron-ui-control ektron-ui-buttonnext">
<button aria-disabled="false" role="button" class="ui-button ui-widget ui-state-default
ui-corner-all ui-button-text-icon-secondary">
```

```
<span class="ui-button-text">Start</span>
<span class="ui-button-icon-secondary ui-icon ui-icon-play"></span>
</button>
</span>
<span class="ektron-ui-control ektron-ui-button next">
<button aria-disabled="false" role="button" class="ui-button ui-widget ui-state-default
ui-corner-all ui-button-text-icon-secondary">
<span class="ui-button-text">Next</span>
<span class="ui-button-icon-secondary ui-icon ui-icon-seek-next"></span>
</button>
</span>
<span class="ektron-ui-control ektron-ui-button finish">
<button aria-disabled="false" role="button" class="ui-button ui-widget ui-state-default
ui-corner-all ui-button-text-icon-secondary">
<span class="ui-button-text">Finish</span>
<span class="ui-button-icon-secondary ui-icon ui-icon-check"></span>
</button>
</span>
</div>
```

Examples for Wizard

Wizard has the following variations:

- **Wizard**. Comprised of a list of available steps, the form for the current step, and a set of buttons to navigate through the process (such as, **Next** and **Back**).
- **Wizard Step**. A single step within a series of wizard steps.

Wizard example

.aspx

```
<ektronUI:Wizard ID="uxDemo" runat="server">
    <ektronUI:Step ID="uxNameStep" Name="Name" ValidationGroup="Name" runat="server">
        <StepTemplate>
            <asp:Label ID="uxNameLabel" AssociatedControlID="uxName" Text="Your Name"
runat="server" />
            <asp:TextBox ID="uxName" ValidationGroup="Name" runat="server" />
            <asp:RequiredFieldValidator ID="uxNameRequired" ControlToValidate="uxName"
ErrorMessage="A name must be provided." ValidationGroup="Name" runat="server" />
        </StepTemplate>
    </ektronUI:Step>
    <ektronUI:Step ID="uxMessageStep" Name="Message" ValidationGroup="Message"
runat="server">
        <StepTemplate>
            <asp:Label ID="uxMessageLabel" AssociatedControlID="uxMessage" Text="Message
to Yourself" runat="server" /><br />
            <asp:TextBox ID="uxMessage" TextMode="MultiLine" ValidationGroup="Message"
runat="server" /><br />
            <asp:RequiredFieldValidator ID="uxMessageRequired"
ControlToValidate="uxMessage" ErrorMessage="A message must be provided."
ValidationGroup="Message" runat="server" />
        </StepTemplate>
    </ektronUI:Step>
```

```
    <ektronUI:Step ID="uxConfirmStep" Name="Confirm" runat="server">
        <StepTemplate>
            <ektronUI:Message DisplayMode="Warning" Text="You're about to send the
configured message to yourself.  Click the finish button to continue." runat="server" />
        </StepTemplate>
    </ektronUI:Step>
    <ektronUI:Summary ID="uxSummary" Name="Summary" runat="server">
        <StepTemplate>
            <%= uxName.Text %>, <%= uxMessage.Text %>
        </StepTemplate>
    </ektronUI:Summary>
</ektronUI:Wizard>
```

NOTE: .aspx.cs not required.

Wizard Step example

.aspx

```
<ektronUI:Wizard ID="uxDemo" runat="server">
    <ektronUI:Step ID="uxStandard" Name="Standard" OnFinish="uxStandard_Finish"
runat="server">
        <StepTemplate>
            <p>This is a standard step.  The next step is disabled (you can't see
it).</p>
            <asp:CheckBox ID="uxEnableDisabled" Text="Enable next step" runat="server"
/>
            <p>
                The above choice will appear as the status of this step so that the user
                can refer back to it.
            </p>
        </StepTemplate>
    </ektronUI:Step>
    <ektronUI:Step ID="uxDisabled" Name="Disabled" Disabled="true" runat="server">
        <StepTemplate>
            This step is disabled.  It isn't visible on the page or in the list of steps
            at first, but it could be enabled from code-behind to have it appear.
        </StepTemplate>
    </ektronUI:Step>
    <ektronUI:Step ID="uxOptional" Name="Optional" Optional="true" runat="server">
        <StepTemplate>
            This step is optional, which means a user can click on a link in the step
list
            after this step from an earlier step.  Note: if this step included form
elements
            that failed validation, the user would still be stopped here.
        </StepTemplate>
    </ektronUI:Step>
    <ektronUI:Step ID="uxValidated" Name="Validated" ValidationGroup="Validated"
runat="server">
        <StepTemplate>
            This step triggers the validation group "Validated" when the next button is
clicked.
            <asp:DropDownList ID="uxOkay" ValidationGroup="Validated" runat="server">
                <asp:ListItem>Okay?</asp:ListItem>
```

```
                <asp:ListItem>Okay.</asp:ListItem>
            </asp:DropDownList>
            <asp:RequiredFieldValidator ID="uxConfirmRequired"
ControlToValidate="uxOkay" InitialValue="Okay?" Text="Please choose a value from the
dropdown." ValidationGroup="Validated" runat="server" />
        </StepTemplate>
    </ektronUI:Step>
    <ektronUI:Step ID="uxConfirm" Name="Confirm" runat="server">
        <StepTemplate>
            This is another standard step, but it's the last one before the user can
click finish.
        </StepTemplate>
    </ektronUI:Step>
    <ektronUI:Summary ID="uxSummary" Name="Summary" runat="server">
        <StepTemplate>
            This is the summary step.  Once this is reached the user can't navigate
through the wizard any more.
            This is an opportunity to inform the user of the results of the wizard.  A
summary step is not required.
        </StepTemplate>
    </ektronUI:Summary>
</ektronUI:Wizard>
```

.aspx.cs

```
protected void uxStandard_Finish(object sender, WizardCompleteEventArgs e)
{
    // Set the status based on the user's input
    this.uxStandard.Status =
        this.uxEnableDisabled.Checked ?
        "Enable" :
        "Disable";

    // Enable/disable the "Disabled" step depending on user input
    this.uxDisabled.Disabled = !this.uxEnableDisabled.Checked;
}
```

3

Templated server controls

8.50 and higher

The templated server controls (TSC) let you interact with the following Ektron CMS user controls.

- *AccessPoint* on page 2054. Access Workarea content directly from the Web site.
- *ContentView* on page 2057. Displays a single piece or multiple pieces of content retrieved from the Ektron CMS.
- *FormControl* on page 2062. Displays a single HTML Form retrieved from the Ektron CMS. FormControl loads and executes required form validation and submit functionality.
- *MenuView* on page 2063. Displays menu data retrieved from the Ektron CMS.
- *Search* on page 2082
  - ProductSearch. Lets site visitors search your website for products.
  - SiteSearch. Lets site visitors search your website for content.
  - UserSearch. Lets site visitors search your website for users.
  - XmlSearch. Lets site visitors search Smart Form content on your website.

You can modify the behavior of templated controls by editing their markup. You no longer use Ektron server control properties nor an XSLT to edit the control's behavior. See also: *Modifying the text displayed by templated server controls* on page 2049

You can edit a control's default markup, as well as the markup on any page that contains it.

Customizing a single instance of a templated search server control

1. Open a page in Visual Studio.
2. Drag an InputView or a ResultsView control onto the page. This section uses the SiteSearchResultsView server control as an example.
3. Insert `<ItemTemplate>` tags between the control's opening and closing tags. `<ItemTemplate>` tags instruct the server control to ignore the default properties.

```
<ektron:SiteSearchResultsView ID="SiteSearchInputView1"
  ControllerID="Scontroller" Visible="true" runat="server">
  <ItemTemplate> </ItemTemplate>
</ektron:SiteSearchResultsView>
```

4. Copy the default markup from the control's corresponding .ascx file. Here is a portion of that.

```
<ektron:SiteSearchResultsView ID="SiteSearchInputView1"
  ControllerID="Scontroller" Visible="true" runat="server">
  <ItemTemplate>
    <h1>Search Phrase:<%# Eval("QueryText") %></h1>
    <asp:ListView ID="aspResults" runat="server"
      DataSource='<%# Eval("Results") %>'
```

```
      ItemPlaceholderID="aspPlaceholder">
      <layouttemplate>
        <ul class="results">
          <asp:PlaceHolder ID="aspPlaceholder" runat="server"> |
          </asp:PlaceHolder>
        </ul>
      </layouttemplate>
      <itemtemplate>
        <li class="result ektron-ui-clearfix">
          <h1 class="title">
            <a href="<%# Eval("Url") %>"><%# Eval("Title") %></a>
          </h1>
          <div class="summary">
            <%# Eval("Summary") %>
          </div>
          <span class="url ektron-ui-quiet">
            <%# Eval("Url") %>
          </span>
          <span class="date ektron-ui-quiet">
            <%# Eval("Date") %>
          </span>
        </li>
      </itemTemplate>
    </asp:ListView>
  </Itemtemplate>
</ektron:SiteSearchResultsView>
```

5. Modify the markup to your specifications.

Modifying the default markup

InputView and ResultsView controls have a corresponding .ascx template that contains their default markup. To change a control's default markup, modify its template.

To find a control's template file, open the folder *siteroot*\workarea\FrameworkUI\Templates\Search, and locate the .ascx file that matches the control's name. For example, *siteroot*\workarea\FrameworkUI\Templates\Search\SiteSearchResultsView.ascx.

IMPORTANT: Files stored in the *siteroot*\Workarea folder are overwritten (or deleted) when you upgrade. To avoid problems, back up the files to a folder outside the siteroot\Workarea folder.

NOTE: "Controller"controls only process data. Because they have no UI component, they do not have a template file.

The default .ascx file affects *all* instances of a control on your website. However, you may customize single instances of a control. To do so, see *Customizing a single instance of a templated search server control* on the previous page.

Modifying a templated Control's Markup

Within an `</ItemTemplate>` tag, use an eval statement `<%# Eval() >)` to access Ektron's Search API object and expose its properties to the data binder. For example, `<%# Eval ("QueryText")%>` displays the search term in the search results.

As an example, you can remove the summary from the search results by deleting this line.

```
<div class="summary"><%# Eval("Summary") %></div>
```

Data fields available to an Eval statement

Use an `<%# Eval()>` statement to determine which field properties appear in search results. For example, `<%# Eval("Summary")%>` displays each content item's summary.

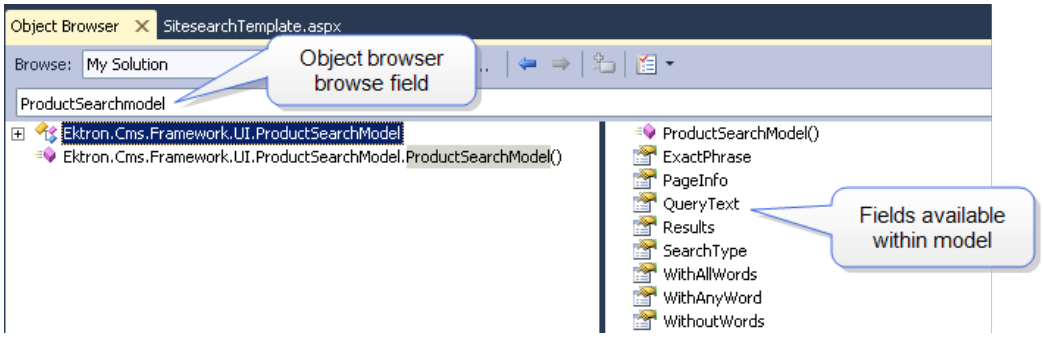
When you insert `<%# Eval()>` between a set of `</ItemTemplate>` tags, a *model* is passed to the statement. Each set of controls has a model that determines available fields and properties. For example, the SearchModel's `Results` property includes these fields:

- Date (late edited)
- Summary
- Title
- Type
- Url

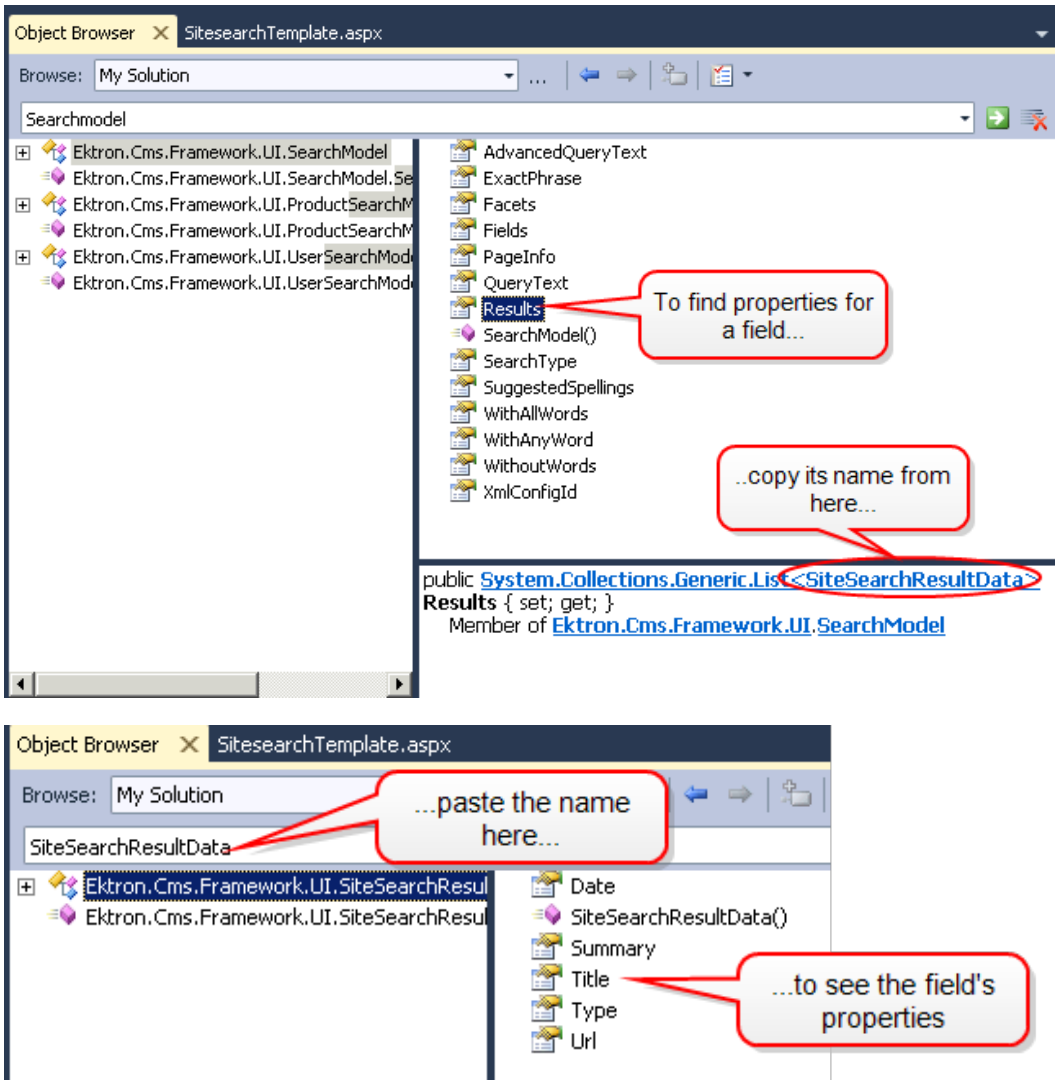
Discovering a model's fields using the object browser

1. Open your Ektron website within Visual Studio.
2. Open the Object Browser.
3. In the **Browse** field, enter the set of controls whose model you want to view. For example, **ProductSearchModel**.

   Other models are **SearchModel** (covers content and XML Smart Forms) and **UserSearchModel**. The model's fields appear in the right pane.



4. To view any field's properties, enter its name into the object browser. Omit the angle brackets (<>).



Modifying the text displayed by templated server controls

To modify the text displayed by a templated control (for example, **no results found**), edit the corresponding .resx file in the *siteroot*\workarea\FrameworkUI\Templates\Search\app_loclaresources folder.

For example, to edit text supplied by the Site Search template, edit *siteroot*\workarea\FrameworkUI\Templates\Search\app_loclaresources\SiteSearchInputView.ascx.resx.

IMPORTANT: Files stored in the *siteroot*\Workarea folder are overwritten (or deleted) when you upgrade. To avoid problems, back up the files to a folder outside the siteroot\Workarea folder.

Injecting your own service

Ektron's templated server controls use an MVC architecture, which is a common design pattern for user interfaces. MVC consists of a

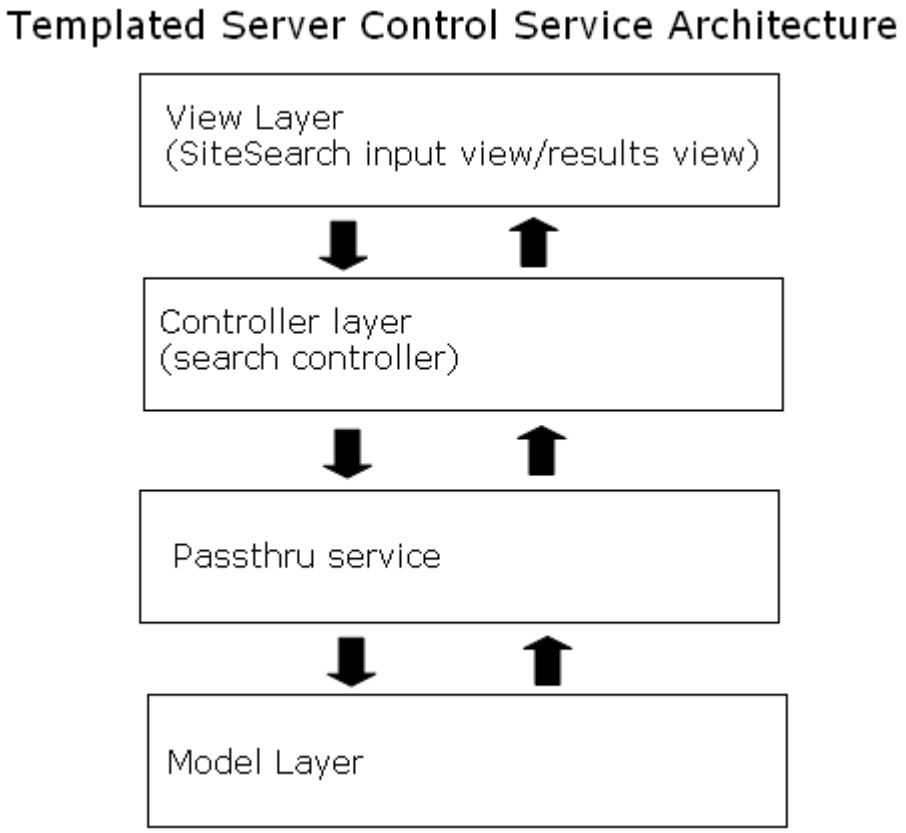
- **M**odel, which stores the state and accesses the application code used by the control; resides in the service layer
- **V**iew, which renders the data
- **C**ontroller, which maps action inputs to the model and elements view; also performs the business logic of the UI

Because MVC architecture separates the display layer from the data layer, a designer can work on the styling of the search results page, while a developer focuses on data being rendered.

Creating a custom service

You can extend the functionality of templated server controls by creating a custom service. For example, you can dynamically replace the content of the search results. You might want to do this to create a profanity filter, or to implement a "blacklist" of replacement terms in search results.

The following image illustrates the location of a custom (passthru) service within MVC architecture.



The passthru service completes these steps.

1. Takes user input from View and Controller Layers.
2. Modifies it.

3. Sends modified data to Model Layer.
4. On the way back, can modify results again.
5. Send results to Controller.
6. Controller pushes results to the View Layer.

Creating a custom service procedure

Use the *siteroot*/`ektron.cms.framework.ui.unity.config` file to map each search type to a service. Below is the section of the file that accomplishes the mapping. By default, Ektron search controls use `ISearch.Service` to take in and return data.

Note separate register statements for regular searches, product searches, and user searches.

NOTE: The ISearchController service handles regular content and XML Smart Form searches.

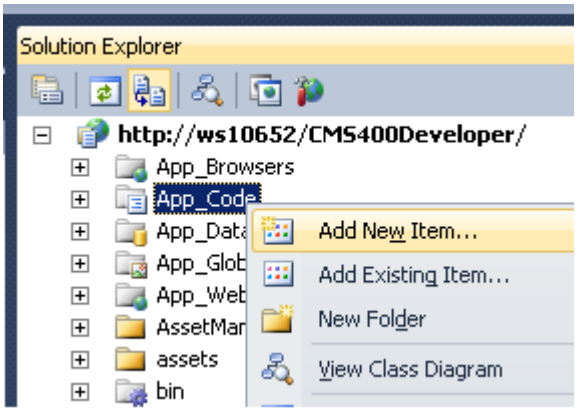
```
<register
  type="Ektron.Cms.Framework.UI.ISearchController, Ektron.Cms.Framework.UI"
  mapTo="Ektron.Cms.Framework.UI.Services.SearchController,
  Ektron.Cms.Framework.UI.Services"/>
<register
  type="Ektron.Cms.Framework.UI.IProductSearchController,
    Ektron.Cms.Framework.UI"
  mapTo="Ektron.Cms.Framework.UI.Services.ProductSearchController,
    Ektron.Cms.Framework.UI.Services"/>
<register
  type="Ektron.Cms.Framework.UI.IUserSearchController,
    Ektron.Cms.Framework.UI"
  mapTo="Ektron.Cms.Framework.UI.Services.UserSearchController,
    Ektron.Cms.Framework.UI.Services"/>
```

Follow these steps to create your own service and map it to a search type.

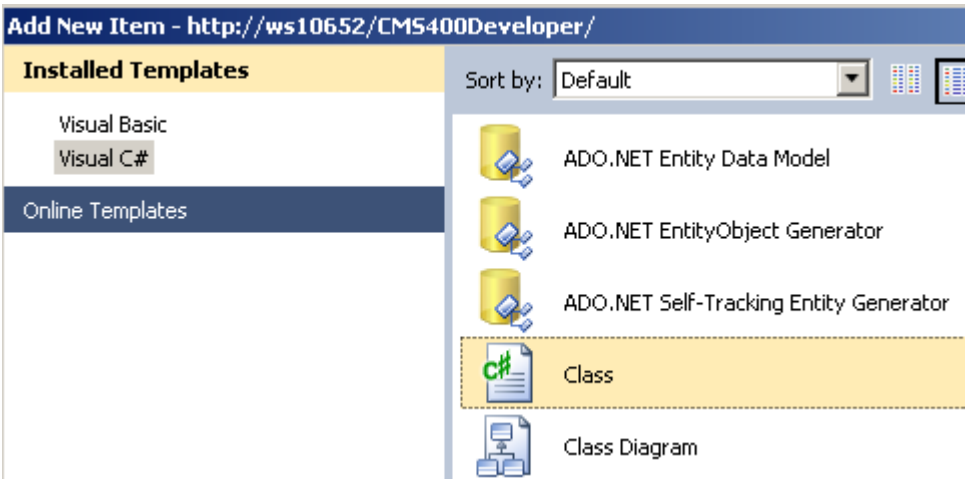
1. In Visual Studio, open *siteroot*/`unity.ui.services.config`.
2. Update the service registration with a new service name. This example assigns to ISearchService a service named DemoSearchService.

   ```
   <register
     type="Ektron.Cms.Framework.UI.ISearchService,
       Ektron.Cms.Framework.UI"
     mapTo="Demo.DemoSearchService"/>
   ```

3. In Solution Explorer, right click the **AppCode** folder and select **Add New Item**.



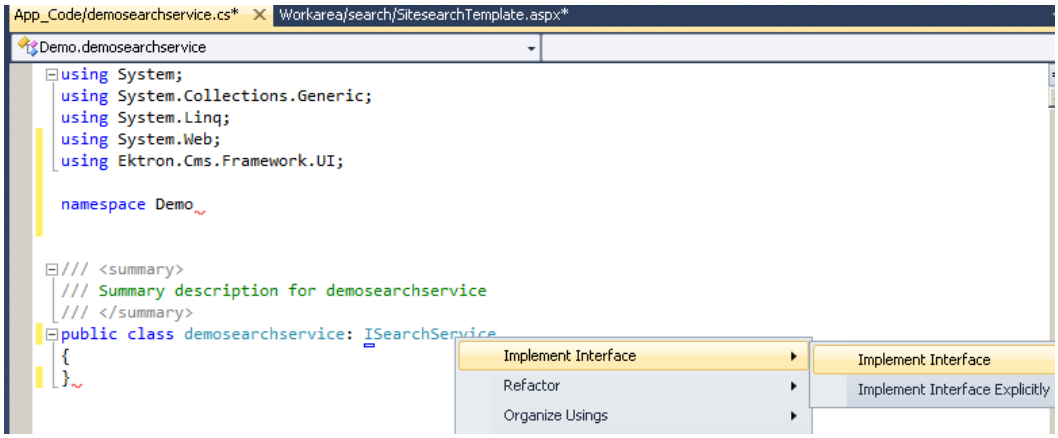
4. Select **C# Class** and give your new service a name.



5. Within the .cs file, insert this statement: `using Ektron.Cms.Framework.UI;`
6. Assign the name of the service that you are overriding. (Service names are listed in `unity.ui.services.config`.)

```
public class DemoSearchService: ISearchService
{
}
```

7. Right click the mouse and select **Implement Interface > Implement Interface**.



8. Several statements are inserted like the following statement, providing inputs to the search service.

```
public void AdvancedSearch(SearchModel model)
{
throw new NotImplementedException();
}
```

9. To create a passthru service, first instantiate the old mock search service. See also: *Creating a custom service* on page 2050

```
public class demosearchservice : ISearchService
ISearchService searchService;
public DemoSearchService()
{
this.searchService= new MockSearchService();
}
```

10. Inside the Advanced, Basic, GetXmlSearchFieldList and XmlSearch methods, call the old service and pass in the same data. For example

```
public void BasicSearch(SearchModel model)
{
this.searchService.BasicSearch(model);
}
```

Now, when you drop inputview and resultsview controls on a page...

- it instantiates the controller.
- it ties the input and results view controls to the controller.
- it instantiates the service, which ties the controller to the service.

Example 1: Any search returns “fifteen” for a result

1. Open the site search template page.
2. Within `SiteSearchResultsView` tags, insert a set of `<itemtemplate>` tags.
3. Within the `<itemtemplate>` tags, insert `<%# Eval(“QueryText”) %>`

```
<ektron:SiteSearchResultsView ID="SiteSearchResultsView1"
  ControllerID="Scontroller" runat="server">
<ItemTemplate> <%# Eval(“QueryText”) %> </ItemTemplate>
</ektron:SiteSearchResultsView \>
```

4. Within the new.cs file that you created in *Creating a custom service procedure* on page 2051, modify the value of `QueryText`. For this example, change the query text to "fifteen." To do this, edit the .cs file, like this.

```
‘Public void BasicSearch(SearchModel model)
{
model.Querytext = “fifteen”; }
```

Example 2: Replace “Summary” with “Ektron” in all results

This example shows how search results may be modified after they are retrieved (that is, on the way back). To do this, edit the .cs file that you created in *Creating a custom service procedure* on page 2051 like this.

```
Public void BasicSearch(SearchModel model)
{
```

```
  model.Results.ForEach(result =>
   { result.Summary = result.Summary.Replace("Summary", "Ektron"); });
}
```

The above code loops through all results (`Results.ForEach`) and replaces "Summary" with "Ektron" in each one.

Example 3: Remove first result

This example shows how to remove the first search result that is retrieved. To do this, edit the .cs file that you created in *Creating a custom service procedure* on page 2051 like this.

```
Public void BasicSearch(SearchModel model)
{
model.Results.RemoveAt(0);
}
```

AccessPoint

8.60 and higher

```
<ektron:AccessPoint>
```

Access Workarea content directly from the Web site. You can assign the access point control to a specific piece of content or a list of content based on taxonomy or collections.

- When assigned to a piece of content, you can edit the content, view the content, or view the content properties.
- When associated with a list of content, you can add items directly to that list.

The Access Point works by itself or in conjunction with other Framework UI controls. You can embed it in a custom template in a templated server control.

Properties for Access Point

- `AddText`. Optional; sets the text to use in place of the default "Add HTML Content" message.
- `ContentType`. Gets or sets the ContentType for the content associated with the control. This will determine what type of content can be added through the access point.
- `DesignTime`. Determines if the control is being rendered in VisualStudio.NET.
- `DisplayType`. Gets or sets the type of menu that is being displayed. For example, Content, Collection, or Folder.
- `FolderId`. Gets or sets the Folder ID where new content will be saved when using the access point. This is used when the DisplayType is Collection or Taxonomy and new content is added.
- `IsPresentationServer`. Gets a flag indicating if the current server is a presentation server.

- `ObjectId`. Gets or sets the Id of the objectthis AccessPoint is associated with. This is directly related to the AccessMenu.DisplayType. If DisplayType is Content, ObjectId should be the Content ID the access point accesses.
- `Title`. Gets or sets the title of the object associated with the access point. This title will be used for display purposes for certain menu types (i.e. content).

Events for Access Point

None.

Methods for Access Point

None.

Theming for Access Point

None.

Examples for Access Point

The Access Point templated server control has the following variations:

- Content
- Collection
- Folder
- Taxonomy

Content example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server">
        <ContentFilters>
            <ektron:ContentFilter Field="Id" Operator="EqualTo" Value="30" />
        </ContentFilters>
    </ektron:ContentModelSource>

    <ektron:AccessPoint ID="accessPoint" runat="server"
        ObjectId="<%# ContentView1.Model.ContentList.First().Id %>"
DisplayType="Content" />
    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource"
        EktronCustomTemplate="Ektron_Default_Template" >
    </ektron:ContentView>
```

NOTE: .aspx.cs not required.

Collection example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server">
        <CollectionFilters>
            <ektron:ContentCollectionFilter Field="Id" Operator="EqualTo" Value="10" />
        </CollectionFilters>
    </ektron:ContentModelSource>

    <ektron:AccessPoint ID="accessPoint" runat="server"
        ObjectId="10" DisplayType="Collection" FolderId="75" />
    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource"
        EktronCustomTemplate="Ektron_ContentList_Template" >
    </ektron:ContentView>
```

NOTE: .aspx.cs not required.

Folder example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server">
        <ContentFilters>
            <ektron:ContentFilter Field="FolderId" Operator="EqualTo" Value="147" />
        </ContentFilters>
    </ektron:ContentModelSource>

    <ektron:AccessPoint ID="accessPoint" runat="server" ObjectId="147"
DisplayType="Folder" />
    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource"
        EktronCustomTemplate="Ektron_ContentList_Template" >
    </ektron:ContentView>
```

NOTE: .aspx.cs not required.

Taxonomy example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server">
        <TaxonomyFilters>
            <ektron:ContentTaxonomyFilter Field="Id" Operator="EqualTo" Value="208" />
        </TaxonomyFilters>
    </ektron:ContentModelSource>

    <ektron:AccessPoint ID="accessPoint" runat="server" ObjectId="208"
DisplayType="Taxonomy"
        FolderId="147" />
    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource"
```

```
        EktronCustomTemplate="Ektron_ContentList_Template" >
    </ektron:ContentView>
```

NOTE: .aspx.cs not required.

ContentView

8.60 and higher

```
<ektron:ContentView>
```

and

```
<ektron:ContentModelSource>
```

The Content templated server control displays a single piece or multiple pieces of content retrieved from the Ektron CMS.

The ContentView templated server control is actually 2 controls that work together:

- `ContentModelSource`. Specifies the content you want to retrieve. You can use filters (CollectionFilters, ContentFilters, MetadataFilters, and TaxonomyFilters) to display a subset of the content you retrieve.
- `ContentView`. Specifies how you want to display the content.

The ContentView templated server control has 2 templates for rendering:

- `Ektron_Default_Template`. Displays a single piece of content. If the ContentModelSource returns more than 1 piece of content, the first piece in the list is rendered.
- `Ektron_ContentList_Template`. Displays a list of content items. You also can create your own ModelTemplate to customize the rendering of the content.

Attributes for Content

Properties for Content

Ektron.Cms.Framework.UI.Controls.Views.ContentView

- `EktronTemplateName`. Gets the default name of the template.
- `Model`. Current Model.
- `ModelSourceID`. Gets or sets the ID of the controller that retrieves the data.
- `TemplateVirtualPath`. Get TemplateVirtualPath.

Ektron.Cms.Framework.UI.Controls.ContentModelSource

- `CollectionFilters`. CollectionFilter Controls for the ModelSource control.
- `ContentFilters`. Content Filter Controls for the ModelSource control.
- `MetadataFilters`. CollectionFilter Controls for the ModelSource control.
- `ReturnMetadata`. Boolean to return metadata for the ModelSource control.
- `TaxonomyFilters`. Taxonomy Filter Controls for the ModelSource control.

Events for Content

None.

Methods for Content

None.

Theming for Content

None.

Examples for Content

The Content templated server control has the following variations:

- CollectionFilters
- Content Item
- Content List
- Custom Item Template
- Custom List Template
- MetadataFilters
- Paging
- TaxonomyFilters

CollectionFilters example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server" OrderByField="Title"
        OrderByDirection="Ascending">
        <Paging RecordsPerPage="3" />
        <CollectionFilters>
            <ektron:ContentCollectionFilter Field="Title" Operator="EqualTo" Value="Our
Team" />
        </CollectionFilters>
    </ektron:ContentModelSource>

    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource"
        EktronCustomTemplate="Ektron_ContentList_Template" >
    </ektron:ContentView>
```

NOTE: .aspx.cs not required.

Content Item example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server">
        <ContentFilters>
            <ektron:ContentFilter Field="Id" Operator="EqualTo" Value="30" />
        </ContentFilters>
    </ektron:ContentModelSource>

    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource"
        EktronCustomTemplate="Ektron_Default_Template" >
    </ektron:ContentView>
```

NOTE: .aspx.cs not required.

Content List example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server">
        <Paging RecordsPerPage="3" />
        <ContentFilters>
            <ektron:ContentFilter Field="Path" Operator="StartsWith"
                Value="MainSite/Content/Support/Knowledge Base/" />
        </ContentFilters>
    </ektron:ContentModelSource>

    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource"
        EktronCustomTemplate="Ektron_ContentList_Template" >
    </ektron:ContentView>
```

NOTE: .aspx.cs not required.

Custom Item Template example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server">
        <ContentFilters>
            <ektron:ContentFilter Field="Id" Operator="EqualTo" Value="30" />
        </ContentFilters>
    </ektron:ContentModelSource>

    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource" >
        <ModelTemplate>
            <ektron:AccessPoint ID="accessPoint" runat="server" ObjectId="<%#
ContentView1.Model.ContentList.First().Id %>"
                DisplayType="Content" />
            <div class="ektron-ui-samplecontent">
                <h1><%# ContentView1.Model.ContentList.First().Title %></h1>
```

```
                <h4><%# ContentView1.Model.ContentList.First().EditorFirstName %>
                    <%# ContentView1.Model.ContentList.First().EditorLastName %>
                </h4>

                <div class="ektron-ui-samplecontent-content">
                    <%# ContentView1.Model.ContentList.First().Html %>
                </div>
            </div>
            </ModelTemplate>
    </ektron:ContentView>
```

NOTE: .aspx.cs not required.

Custom List Template example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server"
OrderByField="DateModified" OrderByDirection="Descending">
        <ContentFilters>
            <ektron:ContentFilter Field="Path" Operator="StartsWith"
Value="MainSite/Content/Community/"/>
        </ContentFilters>
    </ektron:ContentModelSource>

    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource" >
        <ModelTemplate>
            <div class="ektron-ui-samplecontent">
                    <h3>Last <%# ContentView1.Model.ContentList.Count%> Posts </h3>
                    <asp:ListView ID="contentList" runat="server" DataSource='<%#
ContentView1.Model.ContentList %>'>
                        <ItemTemplate>
                            <h2><a href='<%# Eval("Quicklink") %>'><%# Eval("Title")
%></a></h2>
                            <h4>in <%# Eval("FolderName") %></h4>
                            <%# Eval("Teaser")%>
                        </ItemTemplate>
                    </asp:ListView>
            </div>
            </ModelTemplate>
    </ektron:ContentView>
```

NOTE: .aspx.cs not required.

MetadataFilters example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server" OrderByField="Title"
        OrderByDirection="Ascending">
        <Paging RecordsPerPage="3" />
        <MetadataFilters>
            <ektron:ContentMetadataFilter TypeName="Keywords" Operator="Contains"
```

```
                Value="software" />
        </MetadataFilters>
    </ektron:ContentModelSource>

    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource"
        EktronCustomTemplate="Ektron_ContentList_Template" >
    </ektron:ContentView>
```

NOTE: .aspx.cs not required.

Paging example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server">
        <Paging RecordsPerPage="2" />
        <ContentFilters>
            <ektron:ContentFilter Field="Path" Operator="StartsWith"
Value="MainSite/Content/Support/Knowledge Base/" />
        </ContentFilters>
    </ektron:ContentModelSource>

    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource" EktronCustomTemplate="Ektron_ContentList_Template" >
    </ektron:ContentView>
    <ektronUI:Pager ID="contentPager" runat="server"
PageableControlID="contentModelSource" />
```

NOTE: .aspx.cs not required.

TaxonomyFilters example

.aspx

```
<ektron:ContentModelSource ID="contentModelSource" runat="server"
OrderByField="DateCreated"
        OrderByDirection="Descending">
        <Paging RecordsPerPage="3" />
        <ContentFilters>
            <ektron:ContentFilter Field="DateCreated" Operator="GreaterThan"
Value="07/17/2010" />
        </ContentFilters>
        <TaxonomyFilters>
            <ektron:ContentTaxonomyFilter Field="Path" Operator="StartsWith"
                Value="\OnTrek Site Navigation\Company" />
        </TaxonomyFilters>
    </ektron:ContentModelSource>

    <ektron:ContentView ID="ContentView1" runat="server"
ModelSourceID="contentModelSource"
        EktronCustomTemplate="Ektron_ContentList_Template" >
    </ektron:ContentView>
```

NOTE: .aspx.cs not required.

FormControl

8.60 and higher

```
<ektron:FormControl>
```

The FormControl templated server control displays a single HTML Form retrieved from the Ektron CMS. FormControl loads and executes required form validation and submit functionality.

Properties for FormControl

- **`DefaultFormId`**. Sets the HTML Form to render if `DynamicParameter` is unspecified or the parameter contains no value.
- **`DynamicParameter`**. Sets the URL `QueryString parameter` (default is `ekfrm`) that is used to dynamically load HTML Form data into this control.

Events for FormControl

None.

Methods for FormControl

None.

Theming for FormControl

None.

Examples for FormControl

FormControl example

.aspx

```
<ektron:FormControlID="formControl"runat="server"
  DefaultFormId="1234"DynamicParameter="ekfrm"/>
```

NOTE: .aspx.cs not required.

MenuView

8.60 and higher

```
<ektron:MenuView>
```

The MenuView templated server control is used to display menu data retrieved from the Ektron CMS. It can display a single list of menu nodes as well as their child menu items. It can also be used to recursively display submenus and build a tree structure.

The MenuView templated server control is actually 2 controls that work together:

- **MenuModelSource.** Specifies the menu data that you want to retrieve.
- **MenuView**. Specifies how you want to display the menu.

Properties for MenuView

- **`TreeFilter`.** Gets the embedded Paging Control for the Model Source control.

Events for MenuView

None.

Methods for MenuView

None.

Theming for MenuView

None.

Examples for MenuView

The MenuView templated server control has the following variations:

- MenuView
- API and Programatic Data
- Custom Template 8.61 and higher
- Declarative Data
- Parent-Child Menu 8.61 and higher
- Smart Menu Behavior 8.61 and higher

MenuView example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MenuViewDemo.aspx.cs"
  Inherits="MenuViewDemo" %>
```

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
    <head id="Head1" runat="server">
        <title>MenuView</title>
    </head>
    <body>
        <form id="form1" runat="server">
            <div>
                <ektron:MenuModelSource ID="menuModelSource" runat="server">
                    <TreeFilter Id="6" Depth="0" />
                </ektron:MenuModelSource>
                <ektron:MenuView ID="menuView" runat="server"
                  ModelSourceID="menuModelSource" />
            </div>
        </form>
    </body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class MenuViewDemo : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

API and programtic data example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MenuViewDemo.aspx.cs"
  Inherits="MenuViewDemo" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
    <head id="Head1" runat="server">
        <title>MenuView - Api And ProgramaticData</title>
    </head>
    <body>
        <form id="form1" runat="server">
            <div>
                <ektron:MenuModelSource ID="menuModelSource" runat="server">
                    <TreeFilter Id="6" Depth="0" />
                </ektron:MenuModelSource>
```

```
                <ektron:MenuView ID="menuView" runat="server"
                  ModelSourceID="menuModelSource">
                </ektron:MenuView>
            </div>
        </form>
    </body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms.Framework.UI.Controls.EktronUI;

public partial class MenuViewDemo : System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
  {
    menuView.Items.Add(new MenuTreeNode { Text = "One (added in code-behind)",
      NavigateUrl = "http://www.ektron.com" });
    menuView.Items.Add(new MenuTreeNode { Text = "Two (added in code-behind)",
      NavigateUrl = "http://www.msn.com" });

    var node = new MenuTreeNode { Text = "three (added in code-behind)",
      NavigateUrl = "http://www.ektron.com" };
    node.Items.Add(new MenuTreeNode { Text = "child three-a
     (added in code-behind)", NavigateUrl = "http://www.ektron.com" });
    menuView.Items.Add(node);
  }
}
```

Custom template example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MenuViewDemo.aspx.cs"
  Inherits="MenuViewDemo" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head id="Head1" runat="server">
    <title>MenuView - Custom Template</title>
    <style type="text/css">
      .Ektron-Site-MenuSample > ul
       {
         list-style: none;
         padding: 0 20px;
         margin: 0;
         float: left;
         width: 29em;
         background: #222;
         font-size: 1.2em;
```

```
        }

        .Ektron-Site-MenuSample ul.subnav li
        {
          float: left;
          margin: 0;
          padding: 0 15px 0 0;
          position: relative; /*--Declare X and Y axis base for sub navigation--*/
        }

        .Ektron-Site-MenuSample ul.subnav li a
        {
          padding: 10px 5px;
          color: #fff;
          display: block;
          text-decoration: none;
          float: left;
        }

        li.menuitem a
        {
          color: #fff !important;
        }

        .Ektron-Site-MenuSample ul.subnav li a:hover
        {
        }

        li.menuitem a:hover
        {
          background: #222 !important;
          color: #4384a5 !important;
        }

        li.menuitem a.active
        {
           color: #4384a5 !important;
        }

        .Ektron-Site-MenuSample ul.subnav li span
        { /*--Drop down trigger styles--*/
          width: 17px;
          height: 35px;
          float: left;
        }

        .Ektron-Site-MenuSample ul.subnav li ul.subnav
        {
          list-style: none;
          position: absolute; /*--Important - Keeps subnav from affecting main navigation
flow--*/
          left: 0;
          top: 35px;
          background: #333;
          margin: 0;
          padding: 0;
          display: none;
```

```
        float: left;
        width: 170px;
        border: 1px solid #111;
      }

            .Ektron-Site-MenuSample ul.subnav li ul.subnav li
            {
                margin: 0;
                padding: 0;
                border-top: 1px solid #252525; /*--Create bevel effect--*/
                border-bottom: 1px solid #444; /*--Create bevel effect--*/
                clear: both;
                width: 170px;
            }

            .Ektron-Site-MenuSample html ul.subnav li ul.subnav li a
            {
                float: left;
                width: 145px;
                background: #333;
                padding-left: 20px;
            }

            .Ektron-Site-MenuSample html ul.subnav li ul.subnav li a:hover
            { /*--Hover effect for subnav links--*/
                background: #222;
            }
        </style>
    </head>
    <body>
        <form id="form1" runat="server">
            <div>
                <ektron:MenuModelSource ID="uxMenuModelSource" runat="server">
                    <TreeFilter Id="6" />
                </ektron:MenuModelSource>
                <div class="Ektron-Site-MenuSample">
                    <ektron:MenuView ID="uxMainMenu" runat="server"
ModelSourceID="uxMenuModelSource">
                        <ListTemplate>
                            <ul id="menunav" runat="server" class="subnav">
                                <asp:PlaceHolder ID="listPlaceholder" runat="server" />
                            </ul>
                        </ListTemplate>
                        <ItemTemplate>
                            <li <%# ((Eval("Type").ToString().ToLower() == "submenu"))
? @" class=""subnav""" : @" class=""menuitem""" %>>
                                <asp:HyperLink ID="nodeLink" runat="server" Text='<%#
Eval("Text") %>' NavigateUrl='<%# Eval("NavigateUrl") %>' />
                                <asp:PlaceHolder ID="itemPlaceholder" runat="server" />
                            </li>
                        </ItemTemplate>
                    </ektron:MenuView>
                </div>

                <ektronUI:JavaScriptBlock ID="menuCodeBlock" runat="server">
                    <ScriptTemplate>
                        if (typeof (Ektron.Controls) == 'undefined') {
```

```
                Ektron.Controls = {};
            }

            Ektron.Controls.SmartMenu = {
                /* properties */
                isSelected: false,
                expandTreeLoop: 0,
                /* methods */

                initMenu: function () {
                    // Default behaviour
                    Ektron.Controls.SmartMenu.Default();

                    // Expand all Submenus which has css class 'expand'
(Regards to folder Associations.)
                    Ektron.Controls.SmartMenu.ExpandMenu();

                    // Expand all Submenus which has selected menu item.
(Regards to URL.)
                    Ektron.Controls.SmartMenu.SelectMenuItem();
                },

                Default: function () {
                    $("ul.subnav").parent().append("<span></span>"); //Only
shows drop down trigger when js is enabled (Adds empty span tag after ul.subnav*)

                    $("ul.subnav li")
                        .hover(function () {
                                //On hover over, add class "subhover"
                                $(this).addClass("subhover");

                                // Drop down the subnav on hover
                                var hoversSubNav = $(this).find("ul.subnav");
                                hoversSubNav.slideUp().stop(true, false);
                                hoversSubNav.slideDown('fast').show();

                            }, function () {    // On Hover Out
                                // On hover out, remove class "subhover"
                                $(this).removeClass("subhover");

                                //When the mouse hovers out of the subnav,
move it back up
                                var hoversOutSubNav = $(this).find
("ul.subnav");
                                hoversOutSubNav.slideDown().stop(true,
false);
                                hoversOutSubNav.slideUp('fast');
                            }
                        );
                },

                ExpandMenu: function () {
                    $('.expand').parents('.subnav').show();
                    $('.expand').next().show();
                },

                SelectMenuItem: function () {
```

```
                                var selecteditem = $("a[itemselected='True']");
                                $(selecteditem).addClass("active");
                            }
                        }

                        Ektron.Controls.SmartMenu.initMenu();
                    </ScriptTemplate>
                </ektronUI:JavaScriptBlock>
            </div>
        </form>
    </body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class MenuViewDemo : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

Declarative data example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MenuViewDemo.aspx.cs"
Inherits="MenuViewDemo" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>MenuView - Declarative Data</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <ektron:MenuModelSource ID="menuModelSource" runat="server" />
        <ektron:MenuView ID="menuView" runat="server" ModelSourceID="menuModelSource" >
            <Items>
                <ektronUI:MenuTreeItem Id="dec0" Text="Yahoo!"
NavigateUrl="http://www.yahoo.com" />
                <ektronUI:MenuTreeItem Id="dec1" Text="Google"
NavigateUrl="http://www.google.com" />
                <ektronUI:MenuTreeItem Id="dec2" Text="MSN"
NavigateUrl="http://www.msn.com" />
                <ektronUI:MenuTreeItem Id="dec3" Text="Ektron"
NavigateUrl="http://www.ektron.com" />
```

```
            </Items>
        </ektron:MenuView>
    </div>
    </form>
</body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class MenuViewDemo : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

Parent-child menu example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MenuViewDemo.aspx.cs"
Inherits="MenuViewDemo" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
    <head id="Head1" runat="server">
        <title>MenuView - Parent-Child Menu</title>
        <style type="text/css">
            .mainMenu,
            .secondMenu
            {
                clear: both;
            }

            .Ektron-Site-MenuSample ul
            {
                list-style: none;
                padding: 0 20px;
                margin: 0;
                float: left;
                width: 30em;
                background: #222;
                font-size: 1.2em;
            }

            .Ektron-Site-MenuSample ul.subnav li
            {
                float: left;
```

```
                margin: 0;
                padding: 0 15px 0 0;
                position: relative;
            }

            .Ektron-Site-MenuSample ul.subnav li a
            {
                padding: 10px 5px;
                display: block;
                text-decoration: none;
                float: left;
            }

            .Ektron-Site-MenuSample li a
            {
                color: #fff;
            }

            .Ektron-Site-MenuSample li.active a
            {
                color: #4384a5 !important;
            }

            .Ektron-Site-MenuSample li a:hover
            {
                background: #222 !important;
                color: #4384a5 !important;
            }

            .secondMenu ul
            {
                background: #333;
                border-top: 1px solid silver;
            }

            .secondMenu li.active a
            {
                color: gray !important;
            }
        </style>
    </head>
    <body>
        <form id="form1" runat="server">
            <div class="Ektron-Site-MenuSample mainMenu">
                <ektron:MenuModelSource ID="uxMainMenuModelSource" runat="server">
                    <TreeFilter Id="39" />
                </ektron:MenuModelSource>

                <ektron:MenuView ID="uxMainMenu" runat="server"
ModelSourceID="uxMainMenuModelSource">
                    <ListTemplate>
                        <ul id="menunav" runat="server" class="subnav">
                            <asp:PlaceHolder ID="listPlaceholder" runat="server" />
                        </ul>
                    </ListTemplate>
                    <ItemTemplate>
                        <li class='<%# ((Eval("Type") != null && Eval("Type").ToString
```

```
().ToLower() == "submenu")) ? @"subnav" : @"menuitem" %> <%# ((bool)Eval("Selected")) ?
@"active" : "" %>'>
                            <asp:HyperLink ID="nodeLink" runat="server" Text='<%# Eval
("Text") %>' expanded='<%# Eval("Expanded") %>'
                                class='<%# (Boolean.Parse(Eval("Expanded").ToString())) ?
@"expand" : @"collapse" %>'
                                NavigateUrl='<%# Eval("NavigateUrl") %>'
itemselected='<%# Eval("Selected") %>' />
                        </li>
                    </ItemTemplate>
                </ektron:MenuView>
            </div>
            <div class="Ektron-Site-MenuSample secondMenu">
                <ektron:MenuModelSource ID="uxSecondaryMenuSource" runat="server">
                </ektron:MenuModelSource>
                <ektron:MenuView ID="uxSecondaryMenuView" runat="server"
ModelSourceID="uxSecondaryMenuSource">
                    <ListTemplate>
                        <ul id="menunav" runat="server" class="subnav">
                            <asp:PlaceHolder ID="listPlaceholder" runat="server" />
                        </ul>
                    </ListTemplate>
                    <ItemTemplate>
                        <li class='<%# ((Eval("Type").ToString().ToLower() == "submenu"))
? @"subnav" : @"menuitem" %> <%# ((bool)Eval("Selected")) ? @"active" : "" %>'>
                            <asp:HyperLink ID="nodeLink" runat="server" Text='<%# Eval
("Text") %>' expanded='<%# Eval("Expanded") %>'
                                class='<%# (Boolean.Parse(Eval("Expanded").ToString())) ?
@"expand" : @"collapse" %>'
                                NavigateUrl='<%# Eval("NavigateUrl") %>'
itemselected='<%# Eval("Selected") %>' />
                        </li>
                    </ItemTemplate>
                </ektron:MenuView>
            </div>
            <div style="clear: both;">
                <br /><br />
                NOTES:
                <ul>
                    <li>
                        This demo requires a menu in the CMS; be sure to set the
MenuModelSource-TreeFilter
                        Id to the proper menu (if using OnTrek, there is a sample menu
with Id 39).
                    </li>
                    <li>
                        To make the secondary menu work when this code is copied to
                        a test page, you will need either change the link for "Framework
                        UI" (under menu "Sample Developer Master Slave Menu")
                        from
"/OnTrek//developer/framework/ui/controls/TemplatedServerControls/MenuView/ParentChildMen
u.aspx?sample=FrameworkUI"
                        to the URL and querystring for your test page, or add a new child
menu
                        with the URL set to the URL and querystring for your test page
(and add
                        any child nodes that you want to show up in the secondary menu).
```

```
                    </li>
                </ul>
            </div>
        </form>
    </body>
</html>
```

.aspx.cs

IMPORTANT: Regarding the following code near the end of this sample:

```
 //LoadData needs to be called to reload Views, but its protected.
 uxSecondaryMenuSource.GetType().GetMethod("LoadData",
System.Reflection.BindingFlags.NonPublic |
System.Reflection.BindingFlags.Instance |
System.Reflection.BindingFlags.FlattenHierarchy).Invoke
(uxSecondaryMenuSource, new object[] { });
```

In 9.10 SP2 and later releases, you must replace it with the following. The new code does not work with 9.10 SP1 or earlier.

```
void MainMenu_MenuItemChanged(object sender, ParentChildMenuEventArgs
menuEventArgs)
{
 //Initialize Menu Source
 uxSecondaryMenuSource.TreeFilter.Id = menuEventArgs.CurrentMenuItemId;
 uxSecondaryMenuSource.TreeFilter.Depth = 1;
 uxSecondaryMenuSource.LoadData();
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.UI.Controls.Views;

public partial class MenuViewDemo : System.Web.UI.Page
{
    public class ParentChildMenuEventArgs : EventArgs
    {
        public long CurrentMenuItemId { get; set; }
    }

    public delegate void MenuItemChangedHandler(object sender, ParentChildMenuEventArgs
menuEventArgs);
    public event MenuItemChangedHandler MenuItemChanged;

    public long MenuId { get; set; }
    public long ContentId { get; set; }
    public ContentData Content;

    protected void Page_Init(object sender, EventArgs e)
    {
        // Content Dynamic Parameter
```

```
        this.SetContentParameter();
    }

    protected void Page_Load(object sender, EventArgs e)
    {
        uxMainMenu.NodeBound += new TreeViewEventHandler(uxMainMenu_NodeBound);
        uxSecondaryMenuView.NodeBound += new TreeViewEventHandler(uxMainMenu_NodeBound);
        MenuItemChanged += new MenuItemChangedHandler(MainMenu_MenuItemChanged);
    }


    private void SetContentParameter()
    {
        //If there is querystring parameter for Content Id, load the current content.
        string ContentParameterId = "id";

        if (!string.IsNullOrEmpty(Request.QueryString[ContentParameterId]))
        {
            this.ContentId = Convert.ToInt64(Request.QueryString[ContentParameterId]);

            if (this.ContentId > 0)
            {
                var contentManager = new ContentManager();
                Content = contentManager.GetItem(this.ContentId, false);
            }
        }
    }

    void uxMainMenu_NodeBound(object sender, TreeViewEventArgs e)
    {
        // If the Navigation URL matches with the Current URL then Select it.
        if (!string.IsNullOrEmpty(e.Item.NavigateUrl))
        {
            if (Page.Request.RawUrl.ToLower().Contains(e.Item.NavigateUrl.ToLower()))
            {
                e.Item.Selected = true;

                //Firing the MenuItemChangedHandler Event
                if (MenuItemChanged != null && ((MenuView)sender).ID == "uxMainMenu")
                {
                    MenuItemChanged(this, new ParentChildMenuEventArgs {
CurrentMenuItemId = Convert.ToInt64(e.Item.Id.Split('|')[0]) });
                }
            }
        }

        // URL ContentId exists in the Menuitems then Select it.
        if (this.ContentId > 0 && !string.IsNullOrEmpty(e.Item.Id))
        {
            if (e.Item.Items != null && e.Item.Items.Any())
            {
                foreach (Ektron.Cms.Framework.UI.Controls.EktronUI.MenuTreeNode x in
e.Item.Items)
                {
                    if (x.ItemId == this.ContentId.ToString())
                    {
                        // You can manipulate menu item properties here.
```

```
                        x.Selected = true;
                        x.Expanded = true;
                        e.Item.Expanded = true;

                        // Firing the MenuItemChangedHandler Event
                        if (MenuItemChanged != null)
                        {
                            MenuItemChanged(this, new ParentChildMenuEventArgs {
CurrentMenuItemId = Convert.ToInt64(x.ItemId) });
                        }
                    }
                    else
                    {
                        x.Selected = false;
                        x.Expanded = false;
                    }
                }
            }
        }

        // Expand SubMenu Based up on the Associated folders.
        if (e.Item.Type != null && e.Item.Type.ToLower() == "submenu")
        {
            var submenudata =
(Ektron.Cms.Framework.UI.Controls.EktronUI.MenuTreeNode)e.Item;
            if (Content != null && submenudata.AssociatedFolders.Contains
(Content.FolderId))
            {
                e.Item.Expanded = true;
                e.Item.Selected = true;
            }
        }
    }


    void MainMenu_MenuItemChanged(object sender, ParentChildMenuEventArgs
menuEventArgs)
    {
        //Initialize Menu Source
        uxSecondaryMenuSource.TreeFilter.Id = menuEventArgs.CurrentMenuItemId;
        uxSecondaryMenuSource.TreeFilter.Depth = 1;
        uxSecondaryMenuSource.LoadData();
    }
}
```

Smart menu behavior example

.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MenuViewDemo.aspx.cs"
Inherits="MenuViewDemo" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
    <head id="Head1" runat="server">
        <title>MenuView - SmartMenu Sample</title>
```

```
        <style type="text/css">

            .Ektron-Site-MenuSample > ul{
                list-style: none;
                padding: 0 20px;
                margin: 0;
                float: left;
                width: 34em;
                background: #222;
                font-size: 1.2em;
            }

            .Ektron-Site-MenuSample ul.subnav li {
                float: left;
                margin: 0;
                padding: 0 15px 0 0;
                position: relative; /*--Declare X and Y axis base for sub navigation--*/
            }
            .Ektron-Site-MenuSample ul.subnav li a{
                padding: 10px 5px;
                color: #fff;
                display: block;
                text-decoration: none;
                float: left;
            }

            .Ektron-Site-MenuSample ul.subnav li a:hover{
                    background: #222  !important;
                    color: #4384a5 !important;
            }
            .Ektron-Site-MenuSample ul.subnav li.active a
            {
                    color: #4384a5 !important;
            }

            .Ektron-Site-MenuSample ul.subnav li span { /*--Drop down trigger styles--*/
                width: 17px;
                height: 35px;
                float: left;
            }

            .Ektron-Site-MenuSample ul.subnav li ul.subnav {
                list-style: none;
                position: absolute; /*--Important - Keeps subnav from affecting main
navigation flow--*/
                left: 0; top: 35px;
                background: #333;
                margin: 0; padding: 0;
                display: none;
                float: left;
                width: 170px;
                border: 1px solid #111;
            }
            .Ektron-Site-MenuSample ul.subnav li ul.subnav li{
                margin: 0; padding: 0;
                border-top: 1px solid #252525; /*--Create bevel effect--*/
                border-bottom: 1px solid #444; /*--Create bevel effect--*/
```

```
                clear: both;
                width: 170px;
            }
            .Ektron-Site-MenuSample html ul.subnav li ul.subnav li a {
                float: left;
                width: 145px;
                background: #333;
                padding-left: 20px;
            }
            .Ektron-Site-MenuSample html ul.subnav li ul.subnav li a:hover { /*--Hover
effect for subnav links--*/
                background: #222;
            }
        </style>
    </head>
    <body>
        <form id="form1" runat="server">
            <div id="Div1" class="Ektron-Site-MenuSample">
                <ektron:MenuModelSource ID="uxSmartMenuModelSource" runat="server">
                    <TreeFilter Id="50" />
                </ektron:MenuModelSource>

                <ektron:MenuView ID="uxMainMenu" runat="server"
ModelSourceID="uxSmartMenuModelSource">
                    <ListTemplate>
                        <asp:Literal ID="ulBeginTag" runat="server" Text="<ul
class='subnav'>" />
                        <asp:PlaceHolder ID="listPlaceholder" runat="server" />
                        <asp:Literal ID="ulEndTag" runat="server" Text="</ul>" />
                    </ListTemplate>
                    <ItemTemplate>
                        <li class='<%# ((Eval("Type") != null && Eval("Type").ToString
().ToLower() == "submenu")) ? @"subnav" : @"menuitem" %> <%# ((bool)Eval("Selected")) ?
@"active" : "" %>'>
                            <asp:HyperLink ID="nodeLink" runat="server" Text='<%# Eval
("Text") %>' expanded='<%# Eval("Expanded") %>'
                                class='<%# (Boolean.Parse(Eval("Expanded").ToString()))
? @"expand" : @"collapse" %>'
                                NavigateUrl='<%# Eval("NavigateUrl") %>'
itemselected='<%# Eval("Selected") %>' />
                            <ektronUI:PlaceHolderControl ID="itemPlaceholder"
runat="server" />
                        </li>
                    </ItemTemplate>
                </ektron:MenuView>
            </div>
            <ektronUI:JavaScriptBlock ID="menuCodeBlock" runat="server">
                <ScriptTemplate>
                    if (typeof (Ektron.Controls) == 'undefined') {
                        Ektron.Controls = {};
                    }

                    Ektron.Controls.SmartMenu = {
                        /* properties */
                        isSelected: false,
                        expandTreeLoop: 0,
                        /* methods */
```

```
                        initMenu: function () {
                            // Default behaviour
                            Ektron.Controls.SmartMenu.Default();

                            // Expand all Submenus which has css class 'expand' (Regards
to folder Associations.)
                            Ektron.Controls.SmartMenu.ExpandMenu();

                            // Expand all Submenus which has selected menu item. (Regards
to URL.)
                            Ektron.Controls.SmartMenu.SelectMenuItem();
                        },

                        Default: function () {
                            $("ul.subnav").parent().append("<span></span>"); //Only shows
drop down trigger when js is enabled (Adds empty span tag after ul.subnav*)

                            $("ul.subnav li")
                                .hover(function () {
                                        //On hover over, add class "subhover"
                                        $(this).addClass("subhover");

                                        // Drop down the subnav on hover
                                        var hoversSubNav = $(this).find("ul.subnav");
                                        hoversSubNav.slideUp().stop(true, false);
                                        hoversSubNav.slideDown('fast').show();

                                    }, function () {    // On Hover Out
                                        // On hover out, remove class "subhover"
                                        $(this).removeClass("subhover");

                                        //When the mouse hovers out of the subnav, move
it back up
                                        var hoversOutSubNav = $(this).find("ul.subnav");
                                        hoversOutSubNav.slideDown().stop(true, false);
                                        hoversOutSubNav.slideUp('fast');
                                    }
                                );
                        },

                        ExpandMenu: function () {
                            $('.expand').parents('.subnav').show();
                            $('.expand').next().show();
                        },

                        SelectMenuItem: function () {
                            var selecteditem = $("a[itemselected='True']");
                            $(selecteditem).addClass("active");
                            selecteditem.parents('.subnav').show();
                        }
                    }

                    Ektron.Controls.SmartMenu.initMenu();
                </ScriptTemplate>
            </ektronUI:JavaScriptBlock>
        </form>
```

```
    </body>
</html>
```

.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Ektron.Cms;
using Ektron.Cms.Framework.UI.Controls.EktronUI.Utilities;
using Ektron.Cms.Framework.UI.Controls.Views;
using Ektron.Cms.Framework.Content;
using Ektron.Cms.Framework.UI.Controls.EktronUI;
using Ektron.Cms.Framework.UI.Tree;

public partial class MenuViewDemo : System.Web.UI.Page
{
    public class SmartMenuEventArgs : EventArgs
    {
        public long CurrentMenuItemId { get; set; }
    }

    public delegate void MenuItemChangedHandler(object sender, SmartMenuEventArgs
menuEventArgs);
    public event MenuItemChangedHandler MenuItemChanged;
    public long MenuId { get; set; }
    public long ContentId { get; set; }
    public ContentData Content;

    protected override void OnInit(EventArgs e)
    {
        // Content Dynamic Parameter
        this.SetContentParameter();

        base.OnInit(e);
    }

    protected override void OnLoad(EventArgs e)
    {
        base.OnLoad(e);

        uxMainMenu.NodeBound += new TreeViewEventHandler(uxMainMenu_NodeBound);
        uxMainMenu.ListBound += uxMainMenu_ListBound;
        MenuItemChanged += new MenuItemChangedHandler(MainMenu_MenuItemChanged);

        // The smart-menu behavior will select a menu-node whose URL matches the
current page,
        // providing visual feedback to the user. We need to demonstrate this on a
single page;
        // adding some menu items to demonstrate smart-menu behavior (in reality, this
menu would
        // either be on each target page, or shared via a masterpage. For this example,
we need
        // to have links that return to this page, but with different querystring
values we can
```

```
        // use to select the relevant menu node. The following few lines wouldn't
normally be needed):

        uxMainMenu.Items.Add(new MenuTreeNode { Text = "One", NavigateUrl =
Request.FilePath + "?option=one" });
        uxMainMenu.Items.Add(new MenuTreeNode { Text = "Two", NavigateUrl =
Request.FilePath + "?option=two" });
        // add child menu items:
        var node = new MenuTreeNode { Text = "Three", Type = "submenu", NavigateUrl =
Request.FilePath + "?option=three" };
        node.Items.Add(new MenuTreeNode { Text = "Three-A", NavigateUrl =
Request.FilePath + "?option=3a" });
        node.Items.Add(new MenuTreeNode { Text = "Three-B", NavigateUrl =
Request.FilePath + "?option=3b" });
        uxMainMenu.Items.Add(node);
    }

    protected void uxMainMenu_ListBound(object sender, TreeViewEventArgs e)
    {
        var control = sender as Control;
        var ulBeginTag = ControlUtilities.FindControl(control, "ulBeginTag");

        if (ulBeginTag != null)
        {
            var literal = ulBeginTag as Literal;

            if (literal != null)
            {
                literal.Text = (e.Item.Type.ToLower() == "menu") ? @"<ul
class=""mainNavigation subnav"">" : @"<ul class=""subnav"">";
            }
        }
    }

    private void SetContentParameter()
    {

        //If there is querystring parameter for Content Id, load the current content.
        string ContentParameterId = "id";

        if (!string.IsNullOrEmpty(Request.QueryString[ContentParameterId]))
        {
            this.ContentId = Convert.ToInt64(Request.QueryString[ContentParameterId]);

            if (this.ContentId > 0)
            {
                var contentManager = new ContentManager();
                Content = contentManager.GetItem(this.ContentId, false);
            }
        }
    }

    protected void uxMainMenu_NodeBound(object sender, TreeViewEventArgs e)
    {
        // If the Navigation URL matches with the Current URL then Select it.
        if (!string.IsNullOrEmpty(e.Item.NavigateUrl))
        {
```

```
            if (Page.Request.RawUrl.ToLower().Contains(e.Item.NavigateUrl.ToLower()))
            {
                e.Item.Selected = true;

                //Firing the MenuItemChangedHandler Event
                if (MenuItemChanged != null)
                {
                    MenuItemChanged(this, new SmartMenuEventArgs { CurrentMenuItemId =
this.ContentId });
                }
            }
        }

        // URL ContentId exists in the Menuitems then Select it.
        if (this.ContentId > 0 && !string.IsNullOrEmpty(e.Item.Id))
        {
            if (e.Item.Items != null && e.Item.Items.Any())
            {
                foreach (Ektron.Cms.Framework.UI.Controls.EktronUI.MenuTreeNode x in
e.Item.Items)
                {
                    if (x.ItemId == this.ContentId.ToString())
                    {
                        // You can manipulate menu item properties here.
                        x.Selected = true;
                        x.Expanded = true;
                        e.Item.Expanded = true;

                        // Firing the MenuItemChangedHandler Event
                        if (MenuItemChanged != null)
                        {
                            MenuItemChanged(this, new SmartMenuEventArgs {
CurrentMenuItemId = Convert.ToInt64(x.ItemId) });
                        }
                    }
                    else
                    {
                        x.Selected = false;
                        x.Expanded = false;
                    }
                }
            }
        }

        // Expand SubMenu Based up on the Associated folders.
        if (e.Item.Type != null && e.Item.Type.ToLower() == "submenu")
        {
            var submenudata =
(Ektron.Cms.Framework.UI.Controls.EktronUI.MenuTreeNode)e.Item;
            if (Content != null && submenudata.AssociatedFolders.Contains
(Content.FolderId))
            {
                e.Item.Expanded = true;
                e.Item.Selected = true;
            }
        }
    }
```

```
    protected void MainMenu_MenuItemChanged(object sender, SmartMenuEventArgs
menuEventArgs)
    {
        // custom code can utilize this event...

    }
}
```

Search

8.50 and higher

There are several types of search templated server controls:

- ProductSearch. Lets site visitors search your website for products.

  IMPORTANT: Ektron has discontinued new development on its eCommerce module. If you have a license to eCommerce, you will continue to receive support, but if you need to upgrade, contact your account manager for options.

- SiteSearch. Lets site visitors search your website for content.
- UserSearch. Lets site visitors search your website for users.
- XmlSearch. Lets site visitors search Smart Form content on your website.

The Product, Site, User, and Xml search templated server controls let you:

- exercise more granular control over each function.
- customize display of input and results using standard CSS markup.
- eliminate XSLTs. Use ASP.NET templates to control markup.
- use several search `InputView` or search `ResultsView` controls on a template.

Search templated server controls are more specialized than their predecessors, which typically used 1 control to retrieve and display results. To provide search functionality, use 3 templated controls.

Each type of search server control consists of three templated controls and has the following relationship:

1. `[type of search]InputView` (input). Accepts user input of search terms.
2. `[type of search]Controller` (process). Takes input from InputView control and retrieves search results.
3. `[type of search]ResultsView` (output). Shows search results; you can modify the results' display, and so on.

The following example shows this relationship using site search controls.

To replicate all of the previous search server controls' functionality, use a combination of templated controls. For example, use the Pager control to manage the paging of search results. Also, use the Advanced Query Text parameter to limit results to those in a selected folder, in a selected taxonomy category, and so on.

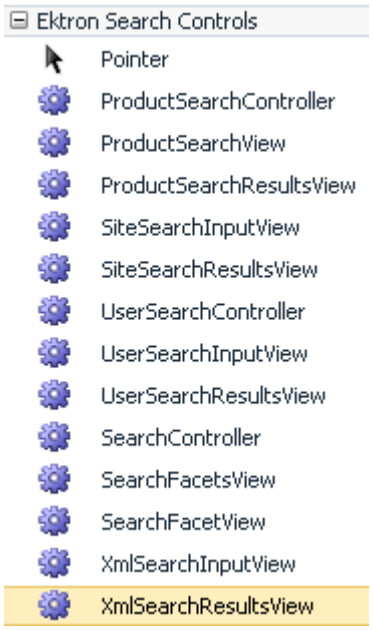
Search server controls have no Ektron-specific properties. In contrast, the pre-8.5 Search server control properties let you configure language, Display XSLT, folder ID, and so on.

References

- ASP.NET MVC Overview
- Building Web Apps without Web Forms
- ASP.NET Data-Bound Web server controls

Adding search server controls to Visual Studio

1. Open your Ektron project in Microsoft Visual Studio 2010.
2. Open the Visual Studio Toolbox.
3. Add a tab and give it a name like `Ektron Search`.
4. Click **Choose Items...**.
5. Click **Browse**.
6. Choose *siteroot*/bin/**Ektron.Cms.Framework.UI.Controls.dll**.
7. Click **OK**. The controls appear on the Toolbox tab.



You should also add EktronUI controls to the Visual Studio toolbox. See *Installing server controls into Visual Studio Toolbox* on page 2135.

Customizing the search behavior with AdvancedQueryText

The AdvancedQueryText parameter lets you customize the behavior of the search controller. As examples, you can restrict search results to

- content properties, such as
  - content in a folder, or a folder and all of its child folders. See also: Recursive Folder Queries.
  - Spanish-language content

- user properties, such as
  - tags set to "Race Car Fans"
  - email address includes "widgets.com"
- eCommerce product properties, such as
  - catalog number is between 1 and 10,000
  - sale price is less than $50 US

The following `AdvancedQueryText` statement restricts search results by language and folder. (It assumes a SiteSearchController named `siteSearchController1` is defined on my page.)

```
<ektron:SiteSearchController ID="siteSearchController1"></ektron:SiteSearchController>
```

Set the `AdvancedQueryText` property in the code-behind for that page. The value assigned to `AdvancedQueryText` is appended to the user's input.

Allowed Operators in the AdvancedQueryText Parameter

- equals (=)
- greater than (>)
- less than (<)
- contains (:)

Retrieving Folder vs. FolderPath

See also: Recursive Folder Queries.

- `folderid`. Identify the folder that contains the content. For example: `folderid:10`

```
protected void Page_Init(object sender, EventArgs e)
  {
    siteSearchController1.AdvancedQueryText = string.Format(
       "({0}:1033 AND {1}:10)",
       Ektron.Cms.Search.SearchContentProperty.Language.Name,
       Ektron.Cms.Search.SearchContentProperty.FolderId.Name);
  }
```

- `folderidpath`. Identify the folder path that contains the content. For example: `folderidpath: "121/71/78/94/10"`

```
protected void Page_Init(object sender, EventArgs e)
  {
    siteSearchController1.AdvancedQueryText = string.Format("({0}={1}) AND {2}:{3}",
    Ektron.Cms.Search.SearchContentProperty.ContentType, 1,
    Ektron.Cms.Search.SearchContentProperty.FolderIdPath, "121/71/78/94/10");
  }
```

NOTE: To exclude a folder but include all of its child folders, use `(folderidpath:"121/71/78/94/10" AND NOT folderid:121)`

Using Solr in Ektron version 9.3

When using the Solr search provider in Ektron version 9.3, the FolderId, FolderPath, and TaxonomyPath `.Contains` syntax acts like a `startswith` expression instead of a `Contains` one. Examples:

FolderIdPath examples

- `FolderIdPath .Contains "72/"` returns content in folder 72 and subfolder 73
- `FolderIdPath .Contains "72/73/"` returns content in folder 73
- `FolderIdPath .Contains "73/", "73", "/73/"` returns no content

Taxonomy examples

- `TaxonomyPath .Contains "12/"` returns content in category 12 and sub-category 13
- `TaxonomyPath .Contains "12/13/"` returns content in sub-category 13
- `TaxonomyPath .Contains "73/", "73", "/73/"` returns no content

Exception:

- `FolderIdPath or TaxonomyPath .Contains "/"` returns content in the root folder/taxonomy only

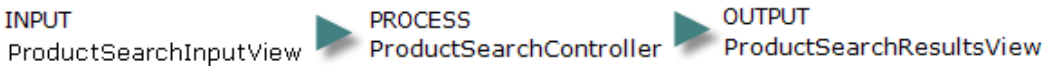
ProductSearch

8.50 and higher

```
<TSC:ProductSearch>
```

IMPORTANT: Ektron has discontinued new development on its eCommerce module. If you have a license to eCommerce, you will continue to receive support, but if you need to upgrade, contact your account manager for options.

The ProductSearch templated server control lets site visitors search your website for products. ProductSearch consists of the following components:



- `ProductSearchInputView`. Accepts user input of search terms.
- `ProductSearchController`. Takes input from `ProductSearchInputView` control and returns search results.
- `ProductSearchResultsView`. Shows search results; you can modify the results' display, paging and so on.

Input and output markup

`InputView` and `ResultsView` controls have a corresponding `.ascx` template that contains default markup. You can modify a template to change a control's default markup. You can find the search control template files at the following locations:

```
[site root]\workarea\FrameworkUI\Templates\Search\ProductSearchInputView.ascx
[site root]\workarea\FrameworkUI\Templates\Search\ProductSearchResultsView.ascx
```

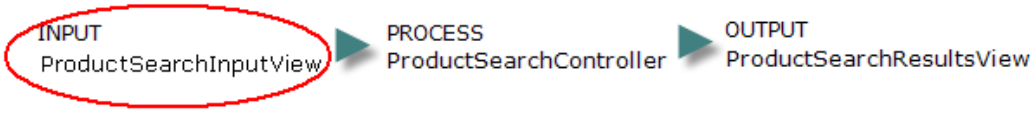
Process controller

The `Controller` control takes input from an `InputView` control, and returns search results to a `ResultsView` control. It provides an interface to the search API,

giving search view controls access to search data. The `Controller` control executes a search using a query string from search `InputView` control's `Text` property, as specified in the code-behind's `OnClick` event. You may modify the `Controller` or write your own.

Unlike the other search controls, the `Controller` control has no UI functionality.

ProductSearchInputView



Use the ProductSearchInputView server control to search eCommerce Products. It provides a Basic and an Advanced search. The query string from the ProductSearchInputView control is passed to the ProductSearchController, which sends results to the ProductSearchResultsView.

- You can place the ProductSearchInputView server control anywhere on a page, and it can be separated from the results display.
- More than 1 ProductSearchInputView server control can reside on a page.
- See also: *Properties for ProductSearchInputView* on page 2089.
- To customize ProductSearchInputView, use the same process outlined in *Customizing the SiteSearchInputView control* on page 2095.

The following image shows the default template for this control.

Click the Advanced link to show these product search filters.

- With all words
- Without words
- Exact Phrase
- With any word

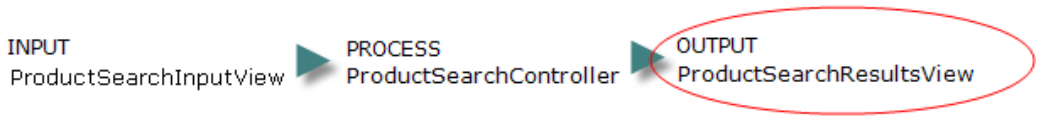
ProductSearchController

The ProductSearchController server control takes input from a ProductSearchInputView control, and returns results to a ProductSearchResultsView control. It provides an interface to the search API, thereby giving SearchView controls access to search data.

- You can modify Ektron's ProductSearchController or write your own.

- The ProductSearchController executes a search using a query string from the ProductSearchInputView control's `Text` property, as specified in the code-behind's `Onclick` event.
- Unlike the other search controls, ProductSearchController has no UI functionality. So, there is no corresponding .ascx file.
- Use only 1 ProductSearchController per page.
- See also: *Properties for ProductSearchController* on page 2089.

ProductSearchResultsView

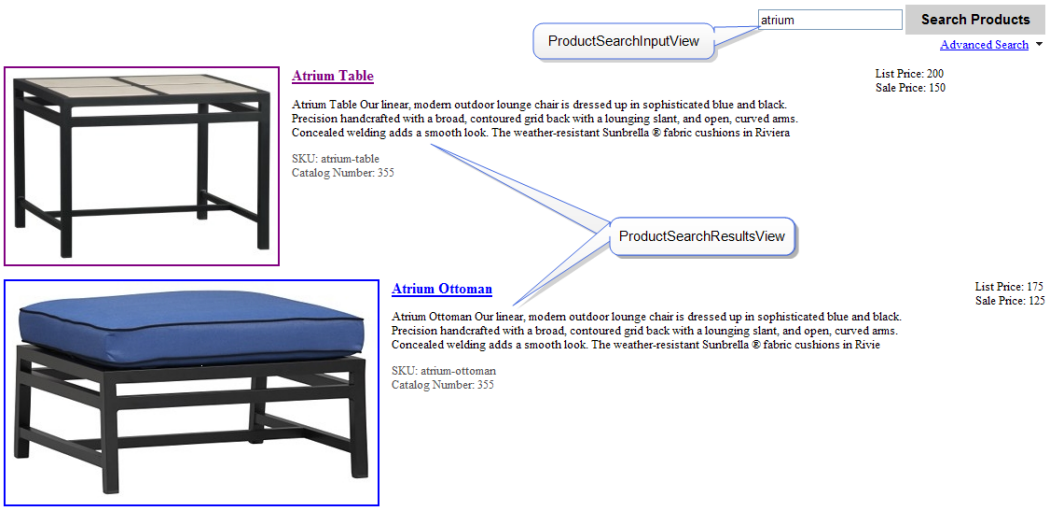


Use the SearchResultsView server control to show search results sent from a ProductSearchController.

The ProductSearchResultsView is tied to the ProductSearchController by setting the property ControllerID.

See also: *Properties for ProductSearchResultsView* on page 2089.

The following image shows the ProductSearchView's search field and button and search results using the ProductSearchResultsView server control's default template. To see an example of displaying user search results, see `[siteroot]\workarea\FrameworkUI\Templates\Search\ProductSearchResultsView.ascx`.



Inserting ProductSearch Templated server controls

To provide search capabilities to your website content, insert all 3 components of the ProductSearch templated server control onto an .aspx template. Because the search functions are managed by separate controls, you can place a search field and button in a one section of a page, and the results in another.

Also, unless you want to use the default paging properties (which show only 10 results), insert an `<TSC:Pager>` control on the page to manage the paging of the search results.

PREREQUISITE

Search server controls were added to the Visual Studio Toolbox. See *Adding search server controls to Visual Studio* on page 2083.

If you drag and drop controls from the Visual Studio Toolbox, the necessary `register` statements are copied from the `web.config` file and added to the page.

```
<add tagPrefix="ektron" namespace="Ektron.Cms.Framework.UI.Controls"
  assembly="Ektron.Cms.Framework.UI.Controls, Version=8.5.0.356,
  Culture=neutral, PublicKeyToken=559a2c4fa21e63be" />
<add tagPrefix="ektronUI"
  namespace="Ektron.Cms.Framework.UI.Controls.EktronUI"
  assembly="Ektron.Cms.Framework.UI.Controls.EktronUI,
  Version=8.5.0.356, Culture=neutral, PublicKeyToken=559a2c4fa21e63be" />
```

If you copy and paste the sample code, add the following `register` statement for templated controls:

```
<%@ Register assembly="Ektron.Cms.Framework.UI.Controls"
  namespace="Ektron.Cms.Framework.UI.Controls"
  tagprefix="ektron" %>
```

For Ektron UI controls, use this `register` statement:

```
<%@ Register assembly="Ektron.Cms.Framework.UI.Controls.EktronUI"
  namespace="Ektron.Cms.Framework.UI.Controls.EktronUI"
  tagprefix="ektronUI" %>
```

The tag prefix can be anything as long as it matches the control's prefix.

The following template shows the 3 components of the ProductSearch templated server control.

```
<%@ Page Language="C#" AutoEventWireup="true"
  CodeFile="TemplatedControl.aspx.cs"
  Inherits="TemplatedControl" ValidateRequest="false" %>

<%@ Register assembly="Ektron.Cms.Framework.UI.Controls"
  namespace="Ektron.Cms.Framework.UI.Controls" tagprefix="ektron" %>

<%@ Register assembly="Ektron.Cms.Framework.UI.Controls.EktronUI"
  namespace="Ektron.Cms.Framework.UI.Controls.EktronUI"
  tagprefix="ektronUI" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
       <title></title>
</head>
  <body>
    <form id="form1" runat="server">
     <div>
       <ektron:ProductSearchInputView ID="ProductSearchInputView1"
          ControllerID="ProductSearchController1" runat="server">
       </ektron:ProductSearchInputView>
```

```
      <ektron:ProductSearchController ID="ProductSearchController1"
        runat="server" />
      <ektron:ProductSearchResultsView
        ControllerID="ProductSearchController1"
        ID="ProductSearchResultsView1" runat="server">
      </ektron:ProductSearchResultsView>
       <ektronUI:Pager ID="pager1" runat="server"
        PageableControlID="ProductSearchController1" ResultsPerPage="10">
      </ektronUI:Pager>
    </div>
   </form>
  </body>
</html>
```

Properties for ProductSearchInputView

- `ControllerID`. The *ID* of the ProductSearchController that provides the data.

```
<CMS:ProductSearchInputView ID="ProductSearchView1"
  ControllerID="ProductSearchController1" runat="server">
</CMS:ProductSearchInputView>
<CMS:ProductSearchController ID="ProductSearchController1"
  runat="server" />
```

- `ExactPhrase`. Sets an EXACT ORDER relationship among search terms.
- `QueryText`. The string used for searching.
- `SearchType`. 1 of the following:
  - Advanced
  - Basic
  - None
  - XmlSearch
- `WithAllWords`. Sets an AND relationship among search terms.
- `WithAnyWord`. Sets an OR relationship among search terms.
- `WithoutWords`. Sets a NOT relationship among search terms.

Properties for ProductSearchController

- `AdvancedQueryText` (Optional). See *Customizing the search behavior with AdvancedQueryText* on page 2083.
- `ID` (Required). Provides the association for the SiteSearchInputView and SiteSearchResultsView server controls. Here is an example of the ID property with the SiteSearchController.

```
<ektron:SiteSearchController ID="SiteSearchController1"
  runat="server" />
```

Properties for ProductSearchResultsView

- `ControllerID`. The *ID* of the UserSearchController that provides the data.

```
<CMS:ProductSearchResultsView ID="ProductSearchView1"
  ControllerID="ProductSearchController1" runat="server">
</CMS:ProductSearchResultsView>
<CMS:ProductSearchController ID="ProductSearchController1"
  runat="server" />
```

- `PageInfo`. Displays information about search result numbers, such as **Displaying results 11 through 20 of 35**.
  - `CurrentPageIndex`. Current page.
  - `EndCount`. Number of results found.
  - `NumberOfPages`. Number of results divided by results per page.
  - `PageCount`. Number of the page being viewed currently.
  - `ResultCount`. Number of the results returned for the search terms.

    IMPORTANT: The result count is only an estimate. The count becomes more accurate you get closer to the final page.

  - `ResultsPerPage`. Number of results displayed per page.
  - `StartCount`. 1.
- `QueryText`. The string used for searching.
- `Results`. The fields returned of ProductSearchResultsData. Display these inside a ListView or Repeater.
  - Catalog Number
  - ImageUrl
  - ListPrice
  - ProductID
  - SalePrice
  - SKU
  - Summary
  - Title
  - URL
- `SearchType`. 1 of the following:
  - Advanced
  - Basic
  - None
  - XmlSearch
- `State`. 1 of the following:
  - initialized. Search has not been run yet
  - no results. No search results were returned
  - search results. Search results were returned

Events for ProductSearch

None.

Methods for ProductSearch

- `Search(string queryText, string advancedQueryText)`
  - Description: Basic search
    - `queryText`. The text you have in the INPUT text box.

        - `advancedQueryText`. The extra query text that is appended to the end of the `queryText`, which you do not want to show on the INPUT text box.
    - Return Type: None.
- `Search(string withAllWords, string withoutWords, string withAnyWord, string exactPhrase, string advancedQueryText)`
    - Description: Advanced search
        - includes `withAllWords`, `withoutWords`, `withAnyWord` and `exactPhrase`.
        - `advancedQueryText`. The extra query text that is appended to the end of the queryText, which you do not want to show on the INPUT text box.
    - Return Type: None.

Theming for ProductSearch

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div id="uxProductSearchInput" class="ektron-ui-control
  ektron-ui-search ektron-ui-search-products">
  <div class="basicSearch">
    <span class="ektron-ui-control
      ektron-ui-input
      ektron-ui-textField"
      id="uxProductSearchInput_ctl00_0_uxSearchText_0_TextField_0_
        uxTextField_0">
    <input type="text"
      id="uxProductSearchInput_ctl00_0_uxSearchText_0_TextField_0_
        aspInput_0"
      name="uxProductSearchInput$ctl00$ctl00$uxSearchText$TextField$
        aspInput">
    </span>
    <span class="ektron-ui-button">
    <input type="button" autocomplete="off" value="Search Products"
       title=""
       onclick="WebForm_DoPostBackWithOptions
         (new WebForm_PostBackOptions
         ('uxProductSearchInput$ctl00$ctl00$uxBasicSearchButton', '',
         true, '', '', false, true));"
       name="uxProductSearchInput_ctl00_0_uxBasicSearchButton_0_Button"
       id="uxProductSearchInput_ctl00_0_uxBasicSearchButton_0_Button"
       class="ui-button ui-widget ui-state-default ui-corner-all"
       role="button">
    </span>
```

```
      <script type="text/javascript" id="uxProductSearchInput_ctl00_0_
        uxBasicSearchButton_0_uxProductSearchInput_ctl00_0_
        uxBasicSearchButton_0EktronScriptBlockptubr_0">
        <!--//--><![CDATA[//>
        <!--
        Ektron.ready(function(event, eventName)
        {
        $ektron("#uxProductSearchInput_ctl00_0_
          uxBasicSearchButton_0_Button").button();
        });
        //-->
        <!]]>
      </script>
  <div class="toggleAdvancedSearchWrapper">
    <a href=""
      class="toggleAdvancedSearch"
      id="uxProductSearchInput_ctl00_0_aspAdvancedSearchLink_0">
      Advanced Search
    </a>
    <a href=""
      class="toggleAdvancedSearchIcon toggleAdvancedSearch"
      id="uxProductSearchInput_ctl00_0_aspAdvancedSearchIcon_0">
      <span class="ui-icon ui-icon-triangle-1-s"></span>
    </a>
  </div>
</div>

      <script type="text/javascript"
        id="uxProductSearchInput_ctl00_0_uxScriptBlockSearch_0_
          uxProductSearchInput_ctl00_0_uxScriptBlockSearch_
          0EktronScriptBlockvbgsy_0">
          <!--//--><![CDATA[//>
          <!--
          Ektron.ready(function(event, eventName)
          {
            setTimeout("Ektron.Controls.Search.SiteSearch.init(
              { clientId: 'uxProductSearchInput_ctl00_0' })", 0);
          });
          //-->
          <!]]>
      </script>
</div>
<div id="uxProductSearchResults"
  class="ektron-ui-control
    ektron-ui-search
    ektron-ui-search-results
    ektron-ui-search-results-products">
  <ul class="results">
    <li class="result     ektron-ui-clearfix">
      <div class="avatar">
        <a href="http://ws10548b/Products/Training/Developer-Training/">
          <img alt="Developer Training"
            src="http://ws10548b/uploadedImages/MainSite/Content/Store/
              Training/DevThumbnail.png"
            title="Developer Training"
            id="uxProductSearchResults_ctl00_0_aspResults_0_aspAvatar_0">
        </a>
```

```
      </div>
      <div class="resultsInfo">
        <h3 class="title">
          <a href="http://ws10548b/Products/Training/Developer-Training/">
            Developer Training
          </a>
        </h3>
        <div class="summary"></div>
        <ul class="ektron-ui-listStyleNone">
          <li class="sku     ektron-ui-quiet">
            SKU: 9546356754365
          </li>
          <li class="catalogNumber     ektron-ui-quiet">
            Catalog Number: 139
          </li>
        </ul>
      </div>
      <ul class="prices">
        <li><span class="listPriceLabel"
             id="uxProductSearchResults_ctl00_0_aspResults_0_
               aspListPrice_0">
               List Price
            </span>: <span class="listPrice">1300</span></li>
        <li><span class="salePriceLabel"
             id="uxProductSearchResults_ctl00_0_aspResults_0_
             aspSalePrice_0">
             Sale Price
            </span>: <span class="salePrice">1400</span></li>
      </ul>
    </li>
  </ul>
</div>
```

Example for ProductSearch

.aspx

```
<ektron:ProductSearchInputView ID="uxProductSearchInput"
  ControllerID="uxSearchController" runat="server">
</ektron:ProductSearchInputView>
<ektron:ProductSearchResultsView ID="uxProductSearchResults"
  ControllerID="uxSearchController" runat="server">
</ektron:ProductSearchResultsView>
<ektron:ProductSearchController ID="uxSearchController" runat="server">
</ektron:ProductSearchController>
```

SiteSearch

8.50 and higher

```
<TSC:SiteSearch>
```

The SiteSearch templated server control lets site visitors search your website. SiteSearch consists of the following components:

- `SiteSearchInputView`. Accepts user input of search terms.
- `SiteSearchController`. Takes input from `SiteSearchInputView` control and returns search results.
- `SiteSearchResultsView`. Shows search results; you can modify the results' display, paging and so on.

To provide search capabilities for your Web site content, insert all three controls onto an `.aspx` template. Because the search functions are managed by separate controls, you can place a search field and button in one section of a page, and the results in another.

Also, unless you want to use the default paging properties (which show only ten results), insert an `<TSC:Pager>` control on the page to manage the paging of the search results.

Input and output markup

`InputView` and `ResultsView` controls have a corresponding `.ascx` template that contains default markup. You can modify a template to change a control's default markup. You can find the search control template files at the following locations:

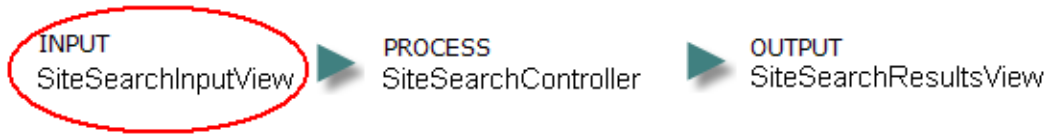
```
[site root]\workarea\FrameworkUI\Templates\Search\SiteSearchInputView.ascx
[site root]\workarea\FrameworkUI\Templates\Search\SiteSearchResultsView.ascx
```

Process controller

The `Controller` control takes input from an `InputView` control, and returns search results to a `ResultsView` control. It provides an interface to the search API, giving search view controls access to search data. The `Controller` control executes a search using a query string from search `InputView` control's `Text` property, as specified in the code-behind's `OnClick` event. You may modify the `Controller` or write your own.

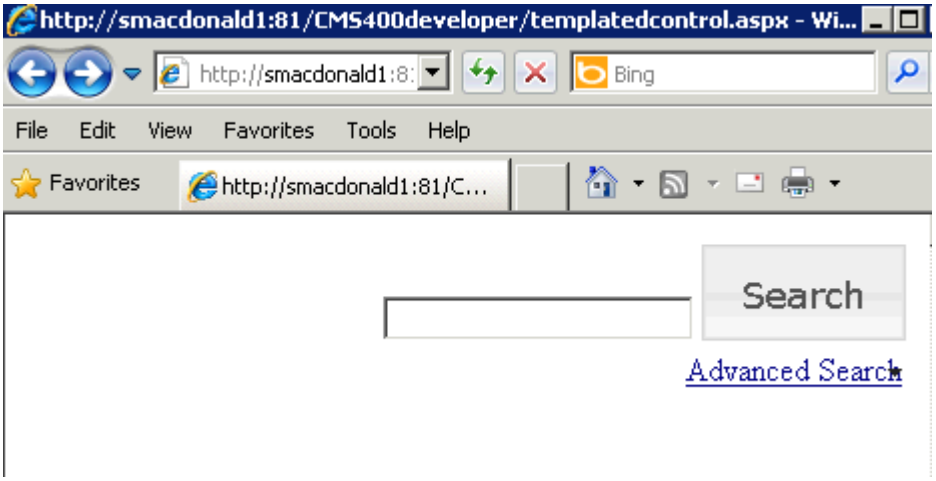
Unlike the other search controls, the `Controller` control has no UI functionality.

SiteSearchInputView



The SiteSearchInputView server control provides a text box and submit button, which visitors use to search content. The control contains no business logic. It merely accepts a search term and passes it to the SiteSearchController, which retrieves and sends results to the SiteSearchResultsView control. See also: *Properties for SiteSearchInputView* on page 2101.

The SiteSearchInputView control's main purpose is a placeholder for markup, which you use to modify its appearance. The control's default markup is very simple: it inserts a text box, a Search button, and an **Advanced Search** link.



To enable search functionality, connect a SiteSearchController to a SiteSearchInputView control. Use the control's `ControllerID` property to do this. As a `value` for the `ControllerID` property, enter the ID of the SiteSearchController that you place within the `<form>` tags. For example:

```
<ektron:SiteSearchInputView ID="SiteSearchInputView1"
  ControllerID="SiteSearchController1" runat="server">
</ektron:SiteSearchInputView>
```

This section also contains the following topics.

Placing a SiteSearchInputView server control on a page

You added search server controls to the Visual Studio Toolbox.

To place a SiteSearchInputView control on a page:

1. Open the page in Visual Studio.
2. Drag and drop the SiteSearchInputView control.

You can place the SiteSearchInputView control anywhere on a page -- it does not need to be next to the results. Also, you can place several SiteSearchInputView controls on a page.

Customizing the SiteSearchInputView control

1. Add a set of `<ItemTemplate>` tags between the SiteSearchInputView control's opening and closing tags. Then, insert an asp text box or an `EktronUI:TextField` to accept the user's input. Here is an example.

```
<ektron:SiteSearchInputView ID="SiteSearchInputView1"
  ControllerID="SiteSearchController1" runat="server">
  <ItemTemplate>
    <ektronUI:TextField ID="queryText" Text='<%# Eval("QueryText") %>'
      runat="server"></ektronUI:TextField>
```

```
  </ItemTemplate>
</ektron:SiteSearchInputView>
```

NOTE: As an alternative, you can copy the line that inserts a text box from *siteroot*\Workarea\FrameworkUI\Templates\Search\SiteSearchInputview.ascx.

Note that the Eval statement (`<%# Eval("QueryText") %>`) databinds the search term to the textbox. As a result, the term remains in the box after the user clicks the search button.

2. Insert an asp search or `EktronUI:Button`, like this.

```
<ektron:SiteSearchInputView ID="SiteSearchInputView1"
ControllerID="SiteSearchController1" runat="server"> <ItemTemplate>
<ektronUI:TextField ID="queryText" Text='<%# Eval("QueryText") %>'
runat="server"></ektronUI:TextField> <ektronUI:Button ID="Button1"
DisplayMode="button" Text="Button 1" runat="server" OnClick="Button1_Click">
</ektronUI:Button> </ItemTemplate>
</ektron:SiteSearchInputView>
```

NOTE: As an alternative, you can copy the line that inserts a search button from *siteroot*\Workarea\FrameworkUI\Templates\Search\SiteSearchInputview.ascx.

3. Assign to the button an `onclick` event that executes a search. To accomplish this, open the page's code-behind file and add the following code.

```
protected void searchButton_Click(object sender, EventArgs e)
  { SiteSearchController1.Search (queryText.Text); }
```

Now you can customize the control's markup as needed.

See also: MSDN Button class, MSDN Textbox class

Separating the search field from the results display

You can place the search field and button on different parts of the page. To do that, insert 2 SiteSearchInputView controls. In one control, place the search field, like this.

```
<ektron:SiteSearchInputView ID="SiteSearchInputView1"
  ControllerID="Scontroller" Visible="true" runat="server">
  <ItemTemplate>
       <ektronUI:TextField ID="queryText" Text='<%# Eval("QueryText") %>'
          runat="server"></ektronUI:TextField>
  </ItemTemplate>
</ektron:SiteSearchInputView>
```

In the second control, in another section of the page, place the button, like this.

```
<ektron:SiteSearchInputView ID="SiteSearchInputView2"
  ControllerID="Scontroller" Visible="true" runat="server">
  <ItemTemplate>
       <ektronUI:Button ID="Button1" DisplayMode="button" Text="Button 1"
         runat="server" OnClick="Button1_Click">
       </ektronUI:Button>
  </ItemTemplate>
</ektron:SiteSearchInputView>
```

Then, assign an onclick event to the button that executes a search. To accomplish this, open the page's code-behind file and add the following code.

```
protected void searchButton_Click(object sender, EventArgs e)
  {
```

```
    SiteSearchController1.Search (queryText.Text);
  }
```

SiteSearchController

The SiteSearchController server control takes input from a SiteSearchInputView control, and returns results to a SiteSearchResultsView control. It provides an interface to the search API, thereby giving SearchView controls access to search data. See also: *Properties for SiteSearchController* on page 2102.

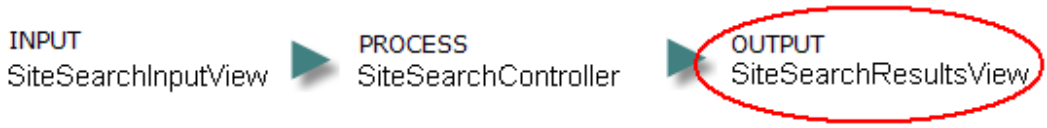
You can modify Ektron's SiteSearchController or write your own.

Use only 1 SiteSearchController per page.

The SiteSearchController executes a search using a query string from the SiteSearchInputView control's `Text` property, as specified in the code-behind's `Onclick` event.

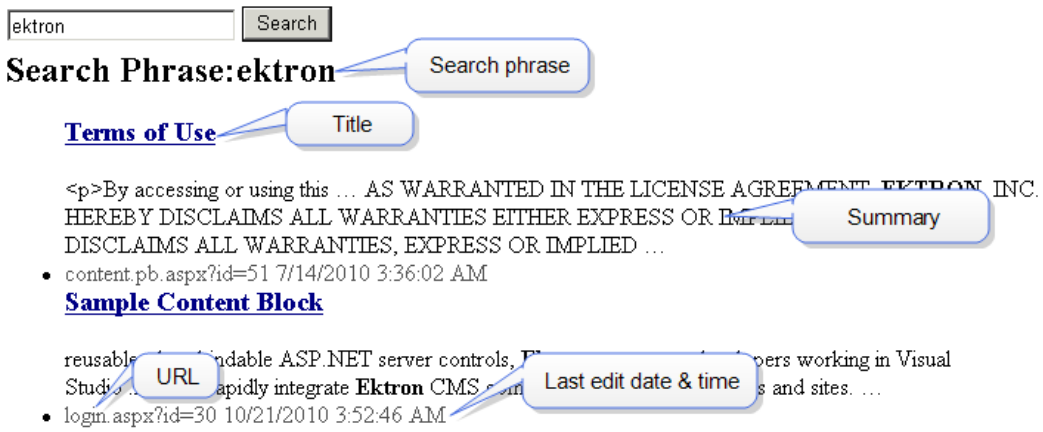
Unlike the other search controls, SiteSearchController has no UI functionality. So, there is no corresponding .ascx file.

SiteSearchResultsView



The SiteSearchResultsView control receives search results from the SiteSearchController and displays them on a page. See also: *Properties for SiteSearchResultsView* on page 2102.

By default, the control's display looks like this.



NOTE: To manage the paging display at the bottom of the page, use an `EktronUI:Pagercontrol`.

To enable search functionality, connect a SiteSearchController to a SiteSearchResultsView control. Use the control's `controllerID` property to do this. For the `controllerID` property value, enter the ID of the SiteSearchController that takes input from SiteSearchInputView control and returns search results to this control. For example:

```
<ektron:SiteSearchResultsView ID="SiteSearchResultsView1"
    ControllerID="SiteSearchController1" runat="server">
</ektron:SiteSearchResultsView>
```

To customize the SiteSearchResultsView control, first add a set of `<ItemTemplate>` tags between the SiteSearchResultsView control's opening and closing tags (as explained in *Templated server controls* on page 2045). Then, modify the search results in the following ways.

NOTE: As an alternative, you can copy the line that inserts a text box from *siteroot*\Workarea\FrameworkUI\Templates\Search\SiteSearchResultsView.ascx.

Displaying the search phrase with results

Use `<%# Eval("QueryText") %>` to databind the search term to the SiteSearchResultsView control. This code sample below displays an example of this **Search Phrase:** followed by the search term.

```
<ektron:SiteSearchResultsView ID="SiteSearchInputView1"
  ControllerID="SiteSearchController1" runat="server">
  <ItemTemplate>
       <p>Search Phrase:<%# Eval("QueryText") %> </p>
  </ItemTemplate>
</ektron:SiteSearchResultsView>
```

Displaying search results using a ListView control

Use a ListView control to display search results from a SiteSearchController.

1. Within a ListView control, use an `<%# Eval("Results") %>` statement to databind search results to the ListView control.

   ```
   <asp:ListView ID="aspResults" runat="server"
     DataSource='<%# Eval("Results") %>'
     ItemPlaceholderID="aspPlaceholder">
   ```

2. To define the main (root) layout of a ListView control, insert a LayoutTemplate. Within that template, insert a placeholder to store the server controls on the page.

   ```
   <asp:ListView ID="aspResults" runat="server"
     DataSource='<%# Eval("Results") %>'
     ItemPlaceholderID="aspPlaceholder">
   <layouttemplate>
     <ul class="results">
       <asp:PlaceHolder ID="aspPlaceholder" runat="server">
       </asp:PlaceHolder>
     </ul>
   </layouttemplate>
   ```

3. Use an ItemTemplate to specify the markup used to generate each record bound to the ListView. Here is an example that displays search results' title, summary, URL and date.

```
<itemtemplate>
  <li>
    <h3><a href="<%# Eval("Url") %>"><%# Eval("Title") %></a></h3>
    <p><%# Eval("Summary") %></p>
    <p><%# Eval("Url") %> <%# Eval("Date") %></p>
  </li>
</itemtemplate>
```

Here is the entire sample.

```
<itemtemplate>
  <asp:ListView ID="aspResults" runat="server"
    DataSource='<%# Eval("Results") %>'
    ItemPlaceholderID="aspPlaceholder">
    <layouttemplate>
       <ul class="results">
         <asp:PlaceHolder ID="aspPlaceholder" runat="server">
         </asp:PlaceHolder></ul>
    </layouttemplate>
    <itemtemplate>
      <li>
         <h3><a href="<%# Eval("Url") %>"><%# Eval("Title") %></a></h3>
         <p><%# Eval("Summary") %></p>
         <p><%# Eval("Url") %> <%# Eval("Date") %></p>
      </li>
    </itemtemplate>
  </asp:ListView>
</itemTemplate>
```

Displaying search results using a GridView control

You can use an `asp:Gridview` control to make search results look like the following.

Search results for: cms

Title	Url	Summary	Date
Controles para Servidor de Ektron CMS400.NET	http://smacdonald1/CMS400Developer/dynamic.aspx?id=30	<ddd/> robust, Ektron <c0>CMS</c0>-powered Web sites and Web applications more rapidly than with other content mana <ddd/> de los datos del <c0>CMS</c0>, expuestos con los controles del servidor va co'digo-behinds. <ddd/> contentos, el <c0>CMS</c0> de Ektron provee de un "ambiente virtual del desarrollo," en contacto en tiempo r <ddd/>	3/24/2006 12:14:05 PM
Sample Content Block	http://smacdonald1/CMS400Developer/dynamic.aspx?id=30	<ddd/> robust, Ektron <c0>CMS</c0>-powered Web sites and Web applications more rapidly than with other content mana <ddd/> can access <c0>CMS</c0> data objects, exposed through the server controls via code-behinds. <ddd/> the Ektron <c0>CMS</c0> provides a "virtual development environment," in real-time contact with the producti <ddd/>	4/14/2006 5:26:42 AM
example_home	http://smacdonald1/CMS400Developer/dynamic.aspx?id=148	<p> Welcome to the Ektron CMS400.NET Developer Section. Ektron has compiled a list of several different examples of developing your Web site with Ektron CMS400.NET. The <ddd/>	1/30/2007 5:26:43 AM

To modify the UI for SearchResultsView:

1. Within the `<ektron:SearchResultsView>` tags, insert `<ItemTemplate></ItemTemplate>`.
2. Between the `<ItemTemplate>` tags, insert the following code, shown in bold.

```
<ektron:SearchResultsView ControllerID="SiteSearchController1"
  runat="server">
  <ItemTemplate>
```

```
    <h2>Search results for: <%# Eval("QueryText") %></h2>
    <asp:Gridview ID="Gridview1" DataSource='<%# Eval("Results") %>'
      runat="server">
    </asp:Gridview>
  </ItemTemplate>
</ektron:SearchResultsView>
```

3. Modify the appearance of the results using Gridview properties and styles. See also: GridView Class.

Inserting SiteSearch templated server controls

To provide search capabilities to your website content, insert all 3 components of the SiteSearch templated server control onto an .aspx template. Because the search functions are managed by separate controls, you can place a search field and button in a one section of a page, and the results in another.

Also, unless you want to use the default paging properties (which show only 10 results), insert an `<TSC:Pager>` control on the page to manage the paging of the search results.

PREREQUISITE

Search server controls were added to the Visual Studio Toolbox. See *Adding search server controls to Visual Studio* on page 2083.

If you drag and drop controls from the Visual Studio Toolbox, the necessary `register` statements are copied from the `web.config` file and added to the page.

```
<add tagPrefix="ektron" namespace="Ektron.Cms.Framework.UI.Controls"
  assembly="Ektron.Cms.Framework.UI.Controls, Version=8.5.0.356,
  Culture=neutral, PublicKeyToken=559a2c4fa21e63be" />
<add tagPrefix="ektronUI"
  namespace="Ektron.Cms.Framework.UI.Controls.EktronUI"
  assembly="Ektron.Cms.Framework.UI.Controls.EktronUI,
  Version=8.5.0.356, Culture=neutral, PublicKeyToken=559a2c4fa21e63be" />
```

If you copy and paste the sample code, add the following `register` statement for templated controls:

```
<%@ Register assembly="Ektron.Cms.Framework.UI.Controls"
  namespace="Ektron.Cms.Framework.UI.Controls"
  tagprefix="ektron" %>
```

For Ektron UI controls, use this `register` statement:

```
<%@ Register assembly="Ektron.Cms.Framework.UI.Controls.EktronUI"
  namespace="Ektron.Cms.Framework.UI.Controls.EktronUI"
  tagprefix="ektronUI" %>
```

The tag prefix can be anything as long as it matches the control's prefix.

The following template shows the 3 components of the SiteSearch templated server control.

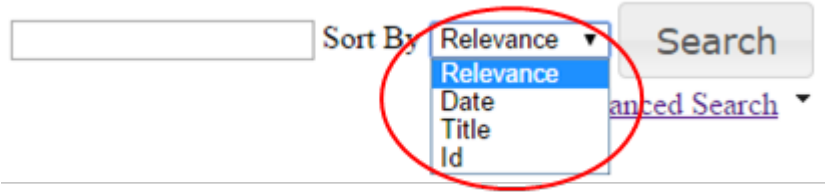
```
<%@ Page Language="C#" AutoEventWireup="true"
  CodeFile="TemplatedControl.aspx.cs"
  Inherits="TemplatedControl" ValidateRequest="false" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
  <head runat="server">
    <title></title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
        <ektron:SiteSearchInputView ID="SiteSearchInputView1"
          ControllerID="SiteSearchController1" runat="server">
        </ektron:SiteSearchInputView>
        <ektron:SiteSearchController ID="SiteSearchController1"
          runat="server" />
        <ektron:SiteSearchResultsView ControllerID="SiteSearchController1"
          ID="SiteSearchResultsView1" runat="server">
        </ektron:SiteSearchResultsView>
        <ektronUI:Pager ID="pager1" runat="server"
          PageableControlID="SiteSearchController1"
          ResultsPerPage="10"></ektronUI:Pager>
      </div>
    </form>
  </body>
</html>
```

You can use the properties with `Eval` statements to display aspects of content search results. For example, `<%# Eval("Summary") %>` displays the summary for content being returned by the search.

Properties for SiteSearchInputView

- `ControllerID`. The *ID* of the SiteSearchController that provides the data.
- `ExactPhrase`. Sets an EXACT ORDER relationship among search terms.
- `Fields`. Label, Name, Xpath: Retrieve information from XML Smart Forms. See also:*XmlSearch* on page 2116.
- `QueryText`. The string that the site visitor entered to begin search.
- `SortProperties`. Provides a dropdown list that lets a site visitor sort search results by a list of pre-selected search properties.



The `Name` of the `SearchProperty` (by which you intend to sort search results) must exactly match the search property name in the search index. Sample code:

```
<cc1:SiteSearchInputView ID="SiteSearchInputView1" runat="server"
ControllerID="SiteSearchController1">
<SortProperties>
       <cc1:RankSearchProperty DisplayName="Relevance" />
       <cc1:DateSearchProperty Direction="Descending" Name="ebddatemodified"
DisplayName="Date" />
       <cc1:StringSearchProperty Direction="Descending" Name="exact_title"
DisplayName="Title" />
```

```
      <cc1:IntegerSearchProperty Direction="Ascending" Name="ebicontentid"
DisplayName="Id" />
</SortProperties>
</cc1:SiteSearchInputView>
```

- `WithAllWords`. Sets an AND relationship among search terms.
- `WithAnyWord`. Sets an OR relationship among search terms.
- `WithoutWords`. Sets a NOT relationship among search terms.

Properties for SiteSearchController

- `AdvancedQueryText`. See *Customizing the search behavior with AdvancedQueryText* on page 2083.
- `AutoCompleteMaxCount`. (Solr) the maximum number of results returned.
- `AutoSuggestMaxCount`. (Search Server, FAST) the maximum number of entries in the auto-suggest list.
- `EnableAutoComplete`. (Solr) suggests terms based on the user's input. For example, of you enter `ba`, autocomplete returns **backup**, **banner**, **base**, **basic**, and so on.
- `EnableAutoSuggest`. (Search Server, FAST) suggests similar terms or common refinements that searchers have used in the past which are related to the search term. As a user enters text, possible matches appear.
- `EnableQueryStatistics`. Enable logging of query statistics in Sharepoint based on the clicks on a search result, which are used to build auto-suggestions for a term.
- `ID` (Required). Provides the association for the SiteSearchInputView and SiteSearchResultsView server controls. The following example shows the ID property with the SiteSearchController.

```
<ektron:SiteSearchController ID="SiteSearchController1"
  runat="server" />
```

Properties for SiteSearchResultsView

- `ControllerID`. The *ID* of the SiteSearchController that provides the data.
- `ElapsedTime`. The time required to execute a search.
- `PageInfo`. Display information about search result numbers, such as **Displaying results 11 through 20 of 35**.
  - CurrentPageIndex. Current page.
  - EndCount. Number of results found.
  - NumberOfPages. Number of results divided by results per page.
  - PageCount. Number of the page being viewed currently.
  - ResultCount. Number of the results returned for the search terms.

    IMPORTANT: The result count is only an estimate. The count becomes more accurate you get closer to the final page.

  - ResultsPerPage. Number of results displayed per page.
  - StartCount. 1.

- `Results`. These are fields returned of type SearchResultsData. Display these inside a ListView or Repeater.
  - Date. Last edit date.
  - Summary. Content summary, up to a maximum of 400 characters.
  - Title
  - Url
  - Type: Content, forms, managed assets, and so on.
    For a full list, enter `SiteSearchResultType` into the Object Browser.
- `QueryText`. String that the site visitor entered to begin search.
- `SearchType`. 1 of the following:
  - Advanced
  - Basic
  - None. No search yet.
  - XmlSearch
- `State`. 1 of the following:
  - initialized. Search has not been run yet.
  - no results. No search results were returned.
  - search results. Search results were returned.
- `Suggested Results`. These fields returned of type SearchResultsData for Suggested Results. Display inside a ListView or Repeater. See also: Displaying Suggested Results.
  - Summary. Content summary, a maximum of 400 characters.
  - Title
  - Url
- `Suggested Spellings`. Displays a list of words that are similar to the entered search phrase. For a sample of using this, open *siteroot*`\workarea\FrameworkUI\Templates\Search\SiteSearchResultsView.ascx` and find `Eval("SuggestedSpellings")`.

Events for SiteSearch

None.

Methods for SiteSearch

- `Search(string queryText, string advancedQueryText)`
  - Description: Basic search
    - `queryText`. The text you have in the INPUT text box.
    - `advancedQueryText`. The extra query text that is appended to the end of the `queryText`, which you do not want to show on the INPUT text box.
  - Return Type: None
- `Search(string withAllWords, string withoutWords, string withAnyWord, string exactPhrase, string advancedQueryText)`

- Description: Advanced search
  - includes `withAllWords`, `withoutWords`, `withAnyWord` and `exactPhrase`.
  - `advancedQueryText`. The extra query text that is appended to the end of the queryText, which you do not want to show on the INPUT text box.
- Return Type: None

Theming for SiteSearch

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div id="uxSiteSearchInput"
  class="ektron-ui-control
  ektron-ui-search
  ektron-ui-search-site">
  <div class=basicSearch>
  <span class="ektron-ui-control ektron-ui-input ektron-ui-textField"
    id="uxSiteSearchInput_ctl00_0_uxSearchText_0_TextField_0_
      uxTextField_0">
  <input type="text"
    id="uxSiteSearchInput_ctl00_0_uxSearchText_0_TextField_0_aspInput_0"
    name="uxSiteSearchInput$ctl00$ctl00$uxSearchText$TextField$aspInput">
  </span>
  <span class="ektron-ui-button">
    <input type="button" autocomplete="off" value="Search" title=""
      onclick="WebForm_DoPostBackWithOptions(new WebForm_PostBackOptions
      ('uxSiteSearchInput$ctl00$ctl00$uxBasicSearchButton', '', true,
        '', '', false, true));"
      name="uxSiteSearchInput_ctl00_0_uxBasicSearchButton_0_Button"
      id="uxSiteSearchInput_ctl00_0_uxBasicSearchButton_0_Button"
      class="ui-button ui-widget ui-state-default ui-corner-all"
      role="button">
  </span>
  <script type="text/javascript"
    id="uxSiteSearchInput_ctl00_0_uxBasicSearchButton_0_uxSiteSearchInput_
      ctl00_0_uxBasicSearchButton_0EktronScriptBlockdmvjp_0">
    <!--//-->
    <![CDATA[//>
    <!--
      Ektron.ready(function(event, eventName)
      {
      $ektron
      ("#uxSiteSearchInput_ctl00_0_uxBasicSearchButton_0_Button"
       ).button();
      });
    //--><!]]>
```

```
  </script>
  <div class="toggleAdvancedSearchWrapper">
  <a onclick="return false;" class="toggleAdvancedSearch"
    id="uxSiteSearchInput_ctl00_0_aspAdvancedSearchLink_0" href="">
    Advanced Search
  </a>
  <a onclick="return false;"
    class="toggleAdvancedSearchIcon toggleAdvancedSearch"
    id="uxSiteSearchInput_ctl00_0_aspAdvancedSearchIcon_0" href="">
    <span class="ui-icon ui-icon-triangle-1-s"></span>
  </a>
  </div>
  </div>
  <script type="text/javascript" id="uxSiteSearchInput_ctl00_0_
    uxScriptBlockSearch_0_uxSiteSearchInput_ctl00_0_uxScriptBlockSearch_
    0EktronScriptBlockmknws_0">
    <!--//-->
    <![CDATA[//>
    <!--
      Ektron.ready(function(event, eventName)
      {
        setTimeout("Ektron.Controls.Search.SiteSearch.init(
          { clientId: 'uxSiteSearchInput_ctl00_0' })", 0);
      });
    //--><!]]>
  </script>
</div>
<div id="uxSiteSearchResults"
  class="ektron-ui-control
    ektron-ui-search
    ektron-ui-search-results
    ektron-ui-search-results-site">
  <div class="section suggested-results">
  </div>
  <div class="section no-results">
  </div>
  <div class="section results">
  <ul>
    <li class="result   ektron-ui-clearfix">
      <h3 class="title">
        <a href="http://ws10548b/Clients/Alex-Tishler,-Aveya/">
        Alex Tishler, Aveya
        </a>
      </h3>
      <div class="summary">  "I have recently purchased
        a copy of OnTrek and am very happy with the software.
        It is amazing how many features are available within
        the network. I am also very impressed by the quality
        and the speed of the technical support."
        … Alex Tishler Lead Analyst <strong>Aveya</strong>
        /uploadedImages/MainSite/Content/Clients/Client_Quotes …"
      </div>
        <span class="url   ektron-ui-quiet">
          http://ws10548b/Clients/Alex-Tishler,-Aveya/</span>
        <span class="date   ektron-ui-quiet">9/7/2011 2:47:07 PM</span>
      </li>
    </ul>
```

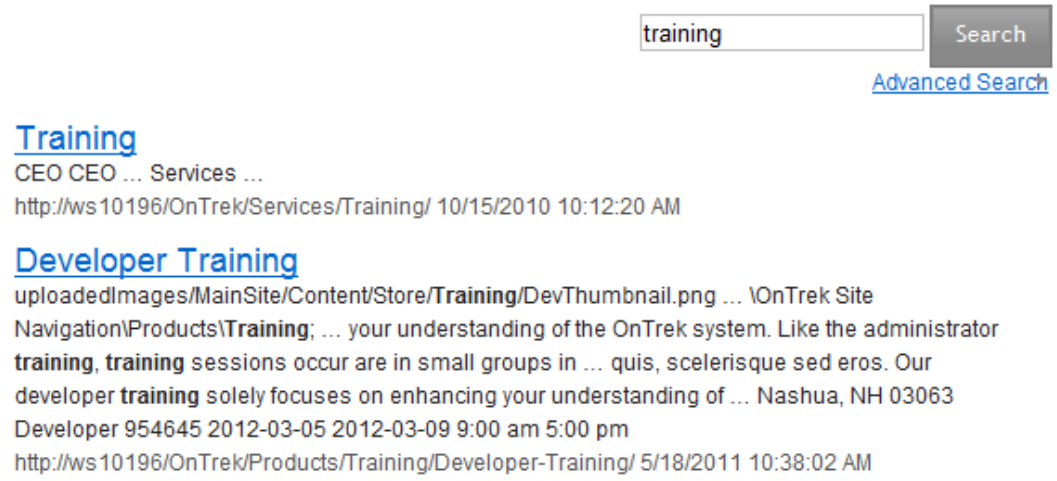
```
  </div>
</div>
```

Properties

None.

Search results

The following example shows just 2 of the many results of searching for "training" in the demo field.



Example for SiteSearch

.aspx

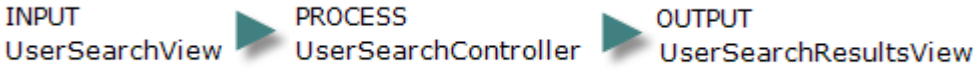
```
<ektron:SiteSearchInputView ID="uxSiteSearchInput"
  ControllerID="uxSearchController" runat="server">
</ektron:SiteSearchInputView>
<ektron:SiteSearchResultsView ID="uxSiteSearchResults"
  ControllerID="uxSearchController" runat="server">
</ektron:SiteSearchResultsView>
<ektron:SiteSearchController ID="uxSearchController" runat="server">
</ektron:SiteSearchController>
```

UserSearch

8.50 and higher

```
<TSC:UserSearch>
```

The UserSearch templated server control lets site visitors search for users and membership users of your website. UserSearch consists of the following components:



- `UserSearchInputView`. Accepts user input of search terms.
- `UserSearchController`. Takes input from `UserSearchInputView` control and returns search results.
- `UserSearchResultsView`. Shows search results; you can modify the results' display, paging and so on.

To provide search capabilities for your Web site content, insert all three controls onto an `.aspx` template. Because the search functions are managed by separate controls, you can place a search field and button in one section of a page, and the results in another.

Also, unless you want to use the default paging properties (which show only ten results), insert an `<TSC:Pager>` control on the page to manage the paging of the search results.

Input and output markup

`InputView` and `ResultsView` controls have a corresponding `.ascx` template that contains default markup. You can modify a template to change a control's default markup. You can find the search control template files at the following locations:

```
[site root]\workarea\FrameworkUI\Templates\Search\UserSearchInputView.ascx
[site root]\workarea\FrameworkUI\Templates\Search\UserSearchResultsView.ascx
```

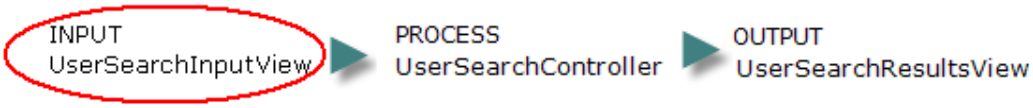
Process controller

The `Controller` control takes input from an `InputView` control, and returns search results to a `ResultsView` control. It provides an interface to the search API, giving search view controls access to search data. The `Controller` control executes a search using a query string from search `InputView` control's `Text` property, as specified in the code-behind's `OnClick` event. You may modify the `Controller` or write your own.

Unlike the other search controls, the `Controller` control has no UI functionality.

Use only one `Controller` control per page.

UserSearchInputView



The UserSearchInputView server control lets site visitors search for Ektron users and members. See also: *Properties for UserSearchInputView* on page 2111, and Membership Users and Groups, and Managing Users and User Groups.

Site visitors can find users using any of these properties:

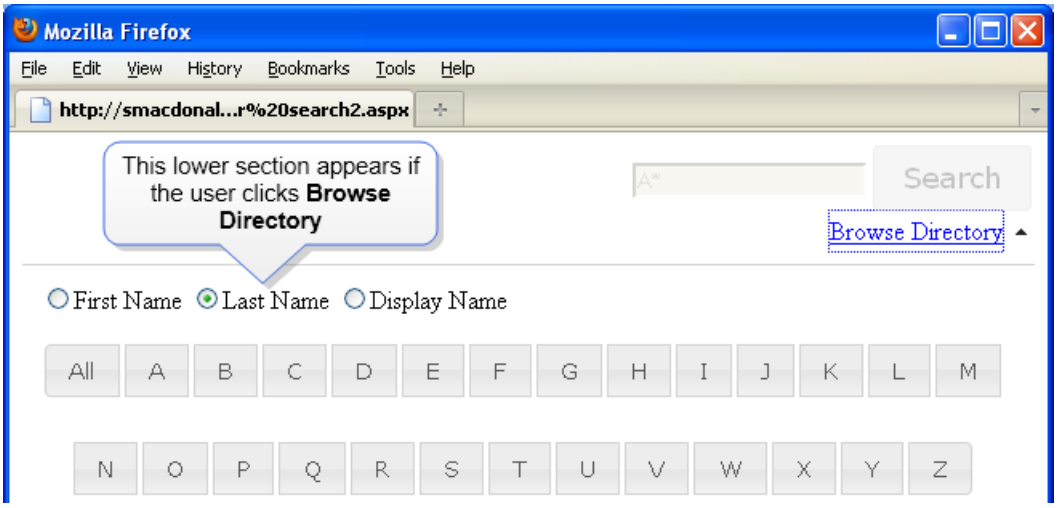
- first name
- last name
- username
- email address

- display name
- tag

Site visitors can use the wildcard character (*) to find all users that fit a certain pattern. For example, to find all users with any property that begins with **adm**, enter **adm***.

To customize the UserSearchInputView control, use the same process as *Customizing the SiteSearchInputView control* on page 2095.

The following image shows the default template's search options.



A site visitor's query string is collected by the UserSearchInputView server control and passed to *ProductSearch* on page 2085 which, in turn, sends results to the *UserSearchResultsView* on the facing page.

You can place the UserSearchInputView server control anywhere on a page. It does not need to be next to its results control. Also, you can place more than one UserSearchInputView control on a page.

UserSearchController

The UserSearchController server control takes input from a UserSearchInputView control, and returns results to a UserSearchResultsView control. It provides an interface to the search API, thereby giving SearchView controls access to search data. See also:*Properties for UserSearchInputView* on page 2111.

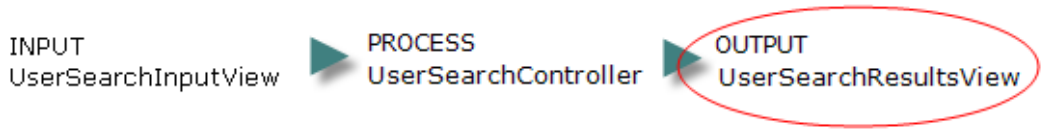
You can modify Ektron's UserSearchController or write your own.

The UserSearchController executes a search using a query string from the UserSearchInputView control's `Text` property, as specified in the code-behind's `Onclick` event.

Unlike the other search controls, UserSearchController has no UI functionality. So, there is no corresponding .ascx file.

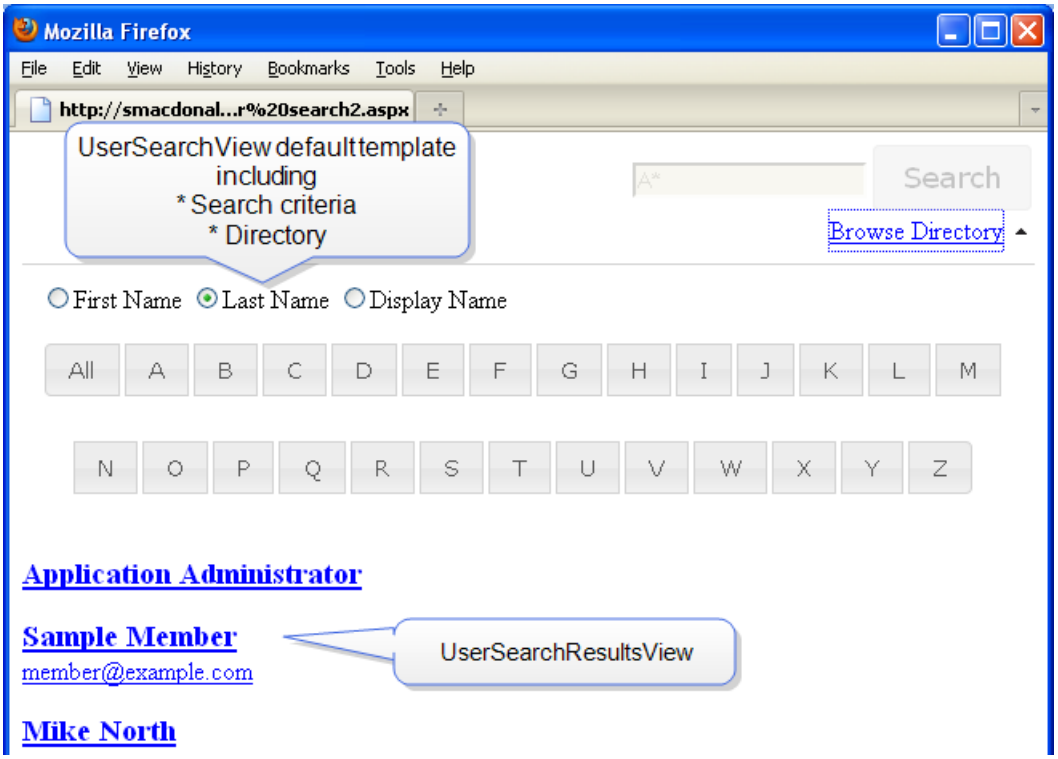
Use only one UserSearchController per page.

UserSearchResultsView



The UserSearchResultsView server control displays the results of a user and member search. The control gets its results from *ProductSearch* on page 2085. See also:*Properties for UserSearchInputView* on page 2111.

The following image shows the UserSearchInputView control's filters, and search results that use the UserSearchResultsView control's default template. To see an example of displaying user search results, see *siteroot*\workarea\FrameworkUI\Templates\Search\UserSearchResultsView.ascx.



To enable the display of user search results, use the UserSearchResultsView control's `controllerID` property to connect it to a SearchController.

For the `ControllerID` property value, enter the `ID` of the UserSearchController that takes input from UserSearchInputView control and returns results to this control. For example:

```
<CMS:UserSearchResultsView ID="UserSearchView1"
  ControllerID="UserSearchController1" runat="server">
</CMS:UserSearchResultsView>
<CMS:UserSearchController ID="UserSearchController1" runat="server" />
```

Inserting UserSearch templated server controls

To provide search capabilities to your website content, insert all 3 components of the UserSearch templated server control onto an .aspx template. Because the search functions are managed by separate controls, you can place a search field and button in a one section of a page, and the results in another.

Also, unless you want to use the default paging properties (which show only 10 results), insert an `<TSC:Pager>` control on the page to manage the paging of the search results.

> Search server controls were added to the Visual Studio Toolbox. See *Adding search server controls to Visual Studio* on page 2083.

If you drag and drop controls from the Visual Studio Toolbox, the necessary `register` statements are copied from the `web.config` file and added to the page.

```
<add tagPrefix="ektron" namespace="Ektron.Cms.Framework.UI.Controls"
  assembly="Ektron.Cms.Framework.UI.Controls, Version=8.5.0.356,
  Culture=neutral, PublicKeyToken=559a2c4fa21e63be" />
<add tagPrefix="ektronUI"
  namespace="Ektron.Cms.Framework.UI.Controls.EktronUI"
  assembly="Ektron.Cms.Framework.UI.Controls.EktronUI,
  Version=8.5.0.356, Culture=neutral, PublicKeyToken=559a2c4fa21e63be" />
```

If you copy and paste the sample code, add the following `register` statement for templated controls:

```
<%@ Register assembly="Ektron.Cms.Framework.UI.Controls"
  namespace="Ektron.Cms.Framework.UI.Controls"
  tagprefix="ektron" %>
```

For Ektron UI controls, use this `register` statement:

```
<%@ Register assembly="Ektron.Cms.Framework.UI.Controls.EktronUI"
  namespace="Ektron.Cms.Framework.UI.Controls.EktronUI"
  tagprefix="ektronUI" %>
```

The tag prefix can be anything as long as it matches the control's prefix.

The following template shows the 3 components of the UserSearch templated server control.

```
<%@ Page Language="C#" AutoEventWireup="true"
  CodeFile="TemplatedControl.aspx.cs"
  Inherits="TemplatedControl" ValidateRequest="false" %>

<%@ Register assembly="Ektron.Cms.Framework.UI.Controls"
  namespace="Ektron.Cms.Framework.UI.Controls" tagprefix="ektron" %>

<%@ Register assembly="Ektron.Cms.Framework.UI.Controls.EktronUI"
  namespace="Ektron.Cms.Framework.UI.Controls.EktronUI"
  tagprefix="ektronUI" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title></title>
  </head>
  <body>
```

```
    <form id="form1" runat="server">
      <div>
        <ektron:UserSearchInputView ID="UserSearchInputView1"
          ControllerID="SearchController1" runat="server">
        </ektron:UserSearchInputView>

        <ektron:UserSearchController ID="SearchController1"
          runat="server" />

        <ektron:UserSearchResultsView ControllerID="SearchController1"
          ID="UserSearchResultsView1" runat="server">
        </ektron:UserSearchResultsView>

        <ektronUI:Pager ID="pager1" runat="server"
          PageableControlID="SiteSearchController1"
          ResultsPerPage="10">
        </ektronUI:Pager>
      </div>
    </form>
  </body>
</html>
```

Properties for UserSearchInputView

- `ControllerID`. ID of the SearchController that provides the data.

  ```
  <ektron:Serena ID="UserSearchInputView1"
    ControllerID="SearchController1" runat="server">
  </ektron:UserSearchInputView>
  ```

- `QueryText`. String that the site visitor entered to begin search.
- `SearchType`. Basic or Advanced search (site visitor clicks **Browse Directory** to access advanced search).

Properties for UserSearchController

- `ID` (Required). Provides the association for the SiteSearchInputView and SiteSearchResultsView server controls. Here is an example of the ID property with the SiteSearchController.

  ```
  <ektron:SiteSearchController ID="SiteSearchController1"
    runat="server" />
  ```

- `AdvancedQueryText` (Optional). See *Customizing the search behavior with AdvancedQueryText* on page 2083.

Properties for UserSearchResultView

- `AvailableFilters`. Label, Name, Value.
- `ControllerID`. The *ID* of the SiteSearchController that provides the data.
- `DirectorySearchLetter`. The letter the site visitor clicks to initiate the search after clicking **Browse Directory**.
- `DirectorySearchType`. First name, last name, or display name.
- `PageInfo`. Display information about search result numbers, such as **Displaying results 11 through 20 of 35**.
  - `CurrentPageIndex`. Current page.
  - `EndCount`. Number of results found.

  - `NumberOfPages`. Number of results divided by results per page.
  - `PageCount`. Number of the page being viewed currently.
  - `ResultCount`. Number of the results returned for the search terms.

    IMPORTANT: The result count is only an estimate. The count becomes more accurate you get closer to the final page.

  - `ResultsPerPage`. Number of results displayed per page.
  - `StartCount`. 1.
- `QueryText`. The string that the site visitor entered to begin search.
- `Results`. Fields returned of UserSearchResults. Display these inside a ListView or Repeater, for example.
  - Avatar
  - DisplayName
  - email
  - FirstName
  - LastName
  - ProfileURL
  - Tags
- `State`. 1 of the following:
  - initialized. Search has not been run yet.
  - no results. No search results were returned.
  - search results. Search results were returned.

Events for UserSearch

None.

Methods for UserSearch

- `AdvancedSearch(List<UserPropertyFilter> filters)`
  - `filters`. Label, Name and Value UserPropertyFilter
- `BasicSearch(string queryText, string advancedQueryText)`
  - `queryText`. The text you have in the INPUT text box.
  - `advancedQueryText`. The extra query text that is appended to the end of the `queryText`, which you do not want to show on the INPUT text box.
- `DirectorySearch(DirectorySearchType type, string advancedQueryText)`
  - type of DirectorySearchType. FirstName, LastName or DisplayName.
  - `advancedQueryText`. The extra query text that is appended to the end of the `queryText`, which you do not want to show on the INPUT text box.
  - Return Type: None.
- `DirectorySearch(DirectorySearchType type, char firstLetter, string advancedQueryText)`

- type of DirectorySearchType. FirstName, LastName or DisplayName.
- `firstLetter`. The first letter of the DirectorySearchType for which you are searching.
- `advancedQueryText`. The extra query text that is appended to the end of the `queryText`, which you do not want to show on the INPUT text box.
- Return Type: None.

- `UpdateFilters(List<UserPropertyFilter> filters)`
  - `filters`. Label, Name and Value UserPropertyFilter

Theming for UserSearch

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div id="uxUserSearchInput"
  class="ektron-ui-control ektron-ui-search ektron-ui-search-users">
  <div class="basicSearch">
    <span class="ektron-ui-control ektron-ui-input ektron-ui-textField"
      id="uxUserSearchInput_ctl00_0_uxSearchText_0_TextField_0_
        uxTextField_0">
      <input type="text"
        id="uxUserSearchInput_ctl00_0_uxSearchText_0_TextField_0_
          aspInput_0"
        value="admin"
        name="uxUserSearchInput$ctl00$ctl00$uxSearchText$
          TextField$aspInput">
    </span>
    <span class="ektron-ui-button">
      <input type="button" autocomplete="off" value="Search" title=""
        onclick="WebForm_DoPostBackWithOptions
          (new WebForm_PostBackOptions
          ('uxUserSearchInput$ctl00$ctl00$uxBasicSearchButton',
          '', true, '', '', false, true));"
        name="uxUserSearchInput_ctl00_0_uxBasicSearchButton_0_Button"
        id="uxUserSearchInput_ctl00_0_uxBasicSearchButton_0_Button"
        class="ui-button ui-widget ui-state-default ui-corner-all"
        role="button">
      </span>
      <script type="text/javascript"
        id="uxUserSearchInput_ctl00_0_uxBasicSearchButton_0_
          uxUserSearchInput_ctl00_0_uxBasicSearchButton_
          0EktronScriptBlockewyld_0">
        <!--//-->
        <![CDATA[//>
        <!--
          Ektron.ready(function(event, eventName)
```

```
        {
        $ektron
          ("#uxUserSearchInput_ctl00_0_uxBasicSearchButton_0_Button"
          ).button();
        });
      //-->
      <![]>
    </script>
  <div class="toggleDirectoryWrapper">
    <a class="toggleDirectorySearch" onclick="return false;"
      id="uxUserSearchInput_ctl00_0_aspDirectorySearchLink_0"
      href="/WorkArea/JavascriptRequired.aspx">Browse Directory</a>
    <a class="toggleDirectorySearchIcon toggleDirectorySearch"
      onclick="return false;"
      id="uxUserSearchInput_ctl00_0_aspDirectorySearchIcon_0"
      href="/WorkArea/JavascriptRequired.aspx">
      <span class="ui-icon ui-icon-triangle-1-s"></span>
    </a>
  </div>
</div>
<div class="directorySearch     ektron-ui-hidden"
  id="uxUserSearchInput_ctl00_0_uxDirectorySearch_0">
  <table id="uxUserSearchInput_ctl00_0_aspDirectorySearchFilters_0">
    <tbody><tr>
    <td><input type="radio" value="firstname"
      name="uxUserSearchInput$ctl00$ctl00$aspDirectorySearchFilters"
      id="uxUserSearchInput_ctl00_0_aspDirectorySearchFilters_0_0_0">
      <label
        for="uxUserSearchInput_ctl00_0_aspDirectorySearchFilters_0_0_0">
        First Name
      </label>
    </td>
    <td><input type="radio" checked="checked" value="lastname"
      name="uxUserSearchInput$ctl00$ctl00$aspDirectorySearchFilters"
      id="uxUserSearchInput_ctl00_0_aspDirectorySearchFilters_0_1_0">
      <label
        for="uxUserSearchInput_ctl00_0_aspDirectorySearchFilters_0_1_0">
        Last Name
      </label>
    </td>
    <td><input type="radio" value="displayname"
      name="uxUserSearchInput$ctl00$ctl00$aspDirectorySearchFilters"
      id="uxUserSearchInput_ctl00_0_aspDirectorySearchFilters_0_2_0">
      <label
        for="uxUserSearchInput_ctl00_0_aspDirectorySearchFilters_0_2_0">
        Display Name
      </label>
    </td>
    </tr>
    </tbody>
  </table>
  <div class="ektron-ui-text-small">
    <div class="ektron-ui-buttonSet ui-buttonset"
      id="uxUserSearchInput_ctl00_0_uxDirectoryButtons_0">
      <script type="text/javascript"
        id="uxUserSearchInput_ctl00_0_uxDirectoryButtons_0_ButtonSet_0_
          uxInitializationScript_0_uxUserSearchInput_ctl00_0_
```

```
            uxDirectoryButtons_0_ButtonSet_0_uxInitializationScript_
            0EktronScriptBlockxmrjj_0">
          <!--//-->
          <![CDATA[//>
          <!--
            Ektron.ready(function(event, eventName)
            {
            Ektron.Controls.EktronUI.ButtonSet.init
              ('#uxUserSearchInput_ctl00_0_uxDirectoryButtons_0');
            });
          //-->
          <!]]>
        </script>
        <input type="button" autocomplete="off" value="All" title=""
          onclick="WebForm_DoPostBackWithOptions
            (new WebForm_PostBackOptions
            ('uxUserSearchInput$ctl00$ctl00$uxDirectoryButtons$btnAll',
            '', true, '', '', false, true));"
            name="uxUserSearchInput_ctl00_0_uxDirectoryButtons_0_
              btnAll_0_Button"
            class="ektron-ui-button ui-button ui-widget ui-state-default
              ui-corner-left"
            id="uxUserSearchInput_ctl00_0_uxDirectoryButtons_0_btnAll_0_
              Button"
            role="button" aria-disabled="false">
        <script type="text/javascript"
          id="uxUserSearchInput_ctl00_0_uxDirectoryButtons_0_btnAll_0_
            uxUserSearchInput_ctl00_0_uxDirectoryButtons_0_btnAll_
            0EktronScriptBlockmnucl_0">
          <!--//-->
          <![CDATA[//>
          <!--
            Ektron.ready(function(event, eventName)
            {
            $ektron
            ("#uxUserSearchInput_ctl00_0_uxDirectoryButtons_0_
              btnAll_0_Button").button();
            });
          //-->
          <!]]>
        </script>
      </div>
    </div>
  </div>
</div>
<div id="uxUserSearchResults"
  class="ektron-ui-control ektron-ui-search ektron-ui-search-results
    ektron-ui-search-results-users">
  <ul class="results">
    <li class="result     ektron-ui-clearfix">
      <div class="avatar">
        <a href="/http://ws10548b/Users/Administrator/">
          <img alt="Administrator"
            src="/uploadedImages/thumb_5baf0_u_MatthewBoone.jpg"
            title="Administrator"
            id="uxUserSearchResults_ctl00_0_aspResults_0_aspImage1_0">
        </a>
```

```
      </div>
      <h3 class="title">
        <a href="http://ws10548b/Users/Administrator/">
          <span class="firstName">Application</span>
          <span class="lastName">Administrator</span>
        </a>
      </h3>
      <span class="email">
      <a href="mailto:admin@ontrek.com">admin@ontrek.com</a></span>
    </li>
  </ul>
</div>
```

Example for UserSearch

.aspx

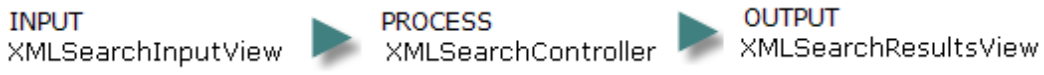
```
<ektron:UserSearchInputView ID="uxUserSearchInput"
 ControllerID="uxSearchController" runat="server">
</ektron:UserSearchInputView>
<ektron:UserSearchResultsView ID="uxUserSearchResults"
  ControllerID="uxSearchController" runat="server">
</ektron:UserSearchResultsView>
<ektron:UserSearchController ID="uxSearchController" runat="server">
</ektron:UserSearchController>
```

XmlSearch

8.50 and higher

```
<TSC:XmlSearch>
```

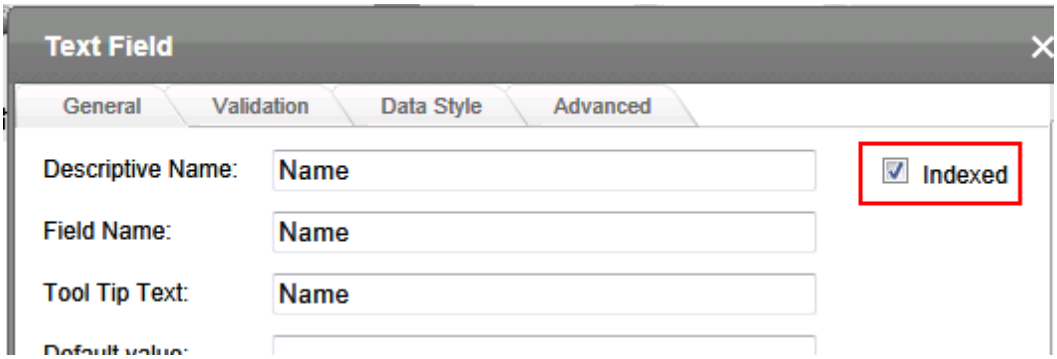
The XMLSearch templated server control displays a Smart Form's fields on a search page. A site visitor can use the fields to search for Smart Form content. XMLSearch consists of the following components:



- `XmlSearchInputView`. Accepts user input of search terms.
- `XmlSearchController`. Takes input from `XmlSearchInputView` control and returns search results.
- `XmlSearchResultsView`. Shows search results; you can modify the results' display, paging and so on.

IMPORTANT: Only fields marked **Indexed** appear on the form (see image below). Also, you cannot search a Smart Form field if its entire xpath exceeds 64 characters. See also: Specifying Which XML Elements are Indexed.

—Image—



To provide search capabilities for your Web site content, insert all three controls onto an .aspx template. Because the search functions are managed by separate controls, you can place a search field and button in one section of a page, and the results in another.

Also, unless you want to use the default paging properties (which show only ten results), insert an <TSC:Pager> server control on the page to manage the paging of the search results.

Input and output markup

InputView and ResultsView controls have a corresponding .ascx template that contains default markup. You can modify a template to change a control's default markup. You can find the search control template files at the following locations:

```
[site root]\workarea\FrameworkUI\Templates\Search\XmlSearchInputView.ascx
[site root]\workarea\FrameworkUI\Templates\Search\XmlSearchResultsView.ascx
```

Process controller

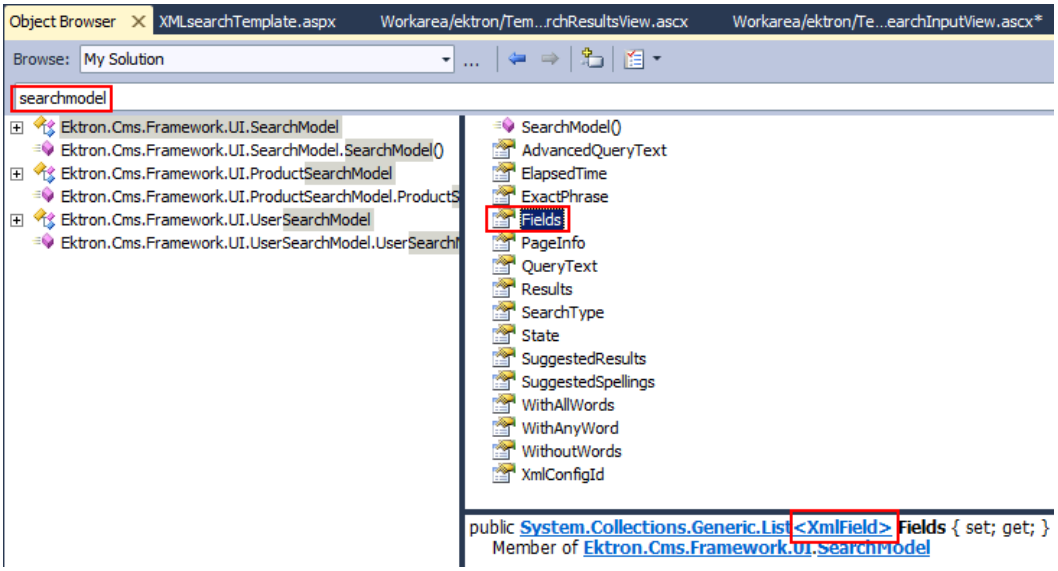
The Controller control takes input from an InputView control, and returns search results to a ResultsView control. It provides an interface to the search API, giving search view controls access to search data. The Controller control executes a search using a query string from search InputView control's Text property, as specified in the code-behind's OnClick event. You may modify the Controller or write your own.

Unlike the other search controls, the Controller control has no UI functionality.

Use only 1 Controller control per page.

Search model for XML Smart Forms

The Search Model gets bound to (that is, determines) the view. When viewing the Search Model in the object browser, you see a property named *Fields*, which is a list of each XML Smart Form field.



To view the Search Model for XML Smart Forms, insert **XMLField** into the Object Browser. The class consists of a...

- **Label**. Friendly name
- **Name**. Internal smart form name
- **Xpath**. Path to field in Smart Form

XML Smart Form search controls use these fields to retrieve and display Smart Form content.

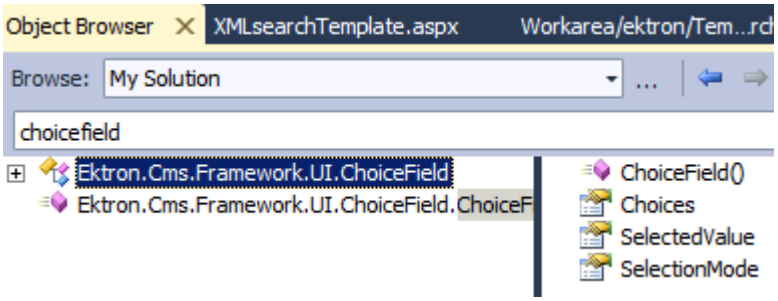
As explained in Using Data Field Types, there are many types of Smart Form fields. Each type has a subclass, which has a corresponding user control, an .ascx template file. User controls are located in `C:\Program Files (x86)\Ektron\CMS400v`*`version`*`\workarea\FrameworkUI\Templates\Search\Fields`. As explained in *Modifying the default markup* on page 2047, to change a control's default markup, modify its template.

IMPORTANT: Files stored in the *`siteroot`*`\Workarea` folder are overwritten (or deleted) when you upgrade. To avoid problems, back up the files to a folder outside the `siteroot\Workarea` folder.

Smart Form field controls receive an XML field and cast it into the specified field type. For example, the Object Browser shows that the ChoiceField template (`ChoiceField.ascx`) contains a

- List of possible choices
- Selected value. The current selection

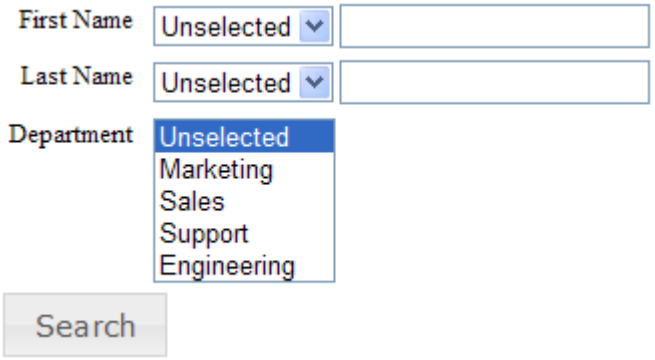
- Selection Mode. Select one or many



The ChoiceField template receives an XML field and casts it into a choice field.

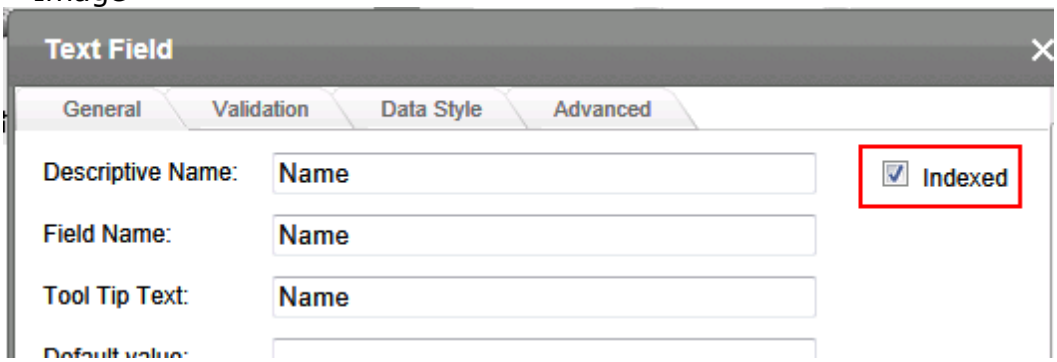
XMLSearchInputView

The XMLSearchInputView server control displays a Smart Form's fields and lets users search on them. See also: *Properties for XmlSearchInputView* on page 2121.



IMPORTANT: Only fields marked **Indexed** appear on the form (see image below). Also, you cannot search a Smart Form field if its entire xpath exceeds 64 characters. See also: Specifying Which XML Elements are Indexed.

—Image—



Because XML fields have a type (for example, string, date, choice, integer, boolean), the template is dynamic and connected to a Smart Form's data structure.

XMLSearchController

The XMLSearchController server control takes input from a XMLSearchInputView control, and returns results to a XMLSearchResultsView control. It provides an interface to the search API, thereby giving SearchView controls access to search data. See also: *Properties for XmlSearchController* on the facing page.

You can modify Ektron's XMLSearchController or write your own.

The XMLSearchController executes a search using a query string from the XMLSearchInputView control's `Text` property, as specified in the code-behind's `Onclick` event.

Unlike the other search controls, XMLSearchController has no UI functionality. So, there is no corresponding .ascx file.

Use only one XMLSearchController per page.

Inserting XmlSearch templated server controls

To provide search capabilities to your website content, insert all 3 components of the XmlSearch templated server control onto an .aspx template. Because the search functions are managed by separate controls, you can place a search field and button in a one section of a page, and the results in another.

Also, unless you want to use the default paging properties (which show only 10 results), insert an `<TSC:Pager>` control on the page to manage the paging of the search results.

> Search server controls were added to the Visual Studio Toolbox. See *Adding search server controls to Visual Studio* on page 2083.

If you drag and drop controls from the Visual Studio Toolbox, the necessary `register` statements are copied from the `web.config` file and added to the page.

```
<add tagPrefix="ektron" namespace="Ektron.Cms.Framework.UI.Controls"
  assembly="Ektron.Cms.Framework.UI.Controls, Version=8.5.0.356,
  Culture=neutral, PublicKeyToken=559a2c4fa21e63be" />
<add tagPrefix="ektronUI"
  namespace="Ektron.Cms.Framework.UI.Controls.EktronUI"
  assembly="Ektron.Cms.Framework.UI.Controls.EktronUI,
  Version=8.5.0.356, Culture=neutral, PublicKeyToken=559a2c4fa21e63be" />
```

If you copy and paste the sample code, add the following `register` statement for templated controls:

```
<%@ Register assembly="Ektron.Cms.Framework.UI.Controls"
  namespace="Ektron.Cms.Framework.UI.Controls"
  tagprefix="ektron" %>
```

For Ektron UI controls, use this `register` statement:

```
<%@ Register assembly="Ektron.Cms.Framework.UI.Controls.EktronUI"
  namespace="Ektron.Cms.Framework.UI.Controls.EktronUI"
  tagprefix="ektronUI" %>
```

The tag prefix can be anything as long as it matches the control's prefix.

The following template shows the 3 components of the XmlSearch templated server control.

```
<%@ Page Language="C#" AutoEventWireup="true"
  CodeFile="TemplatedControl.aspx.cs"
  Inherits="TemplatedControl" ValidateRequest="false" %>

<%@ Register assembly="Ektron.Cms.Framework.UI.Controls"
  namespace="Ektron.Cms.Framework.UI.Controls" tagprefix="ektron" %>

<%@ Register assembly="Ektron.Cms.Framework.UI.Controls.EktronUI"
  namespace="Ektron.Cms.Framework.UI.Controls.EktronUI"
  tagprefix="ektronUI" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
       <title></title>
</head>
  <body>
    <form id="form1" runat="server">
      <div>
        <ektron:XmlSearchInputView ID="XmlSearchInputView1"
          ControllerID="XmlSearchController1" XmlConfigId="9"
          runat="server">
        </ektron:XmlSearchInputView>

        <ektron:XmlSearchController ID="XmlSearchController1"
          runat="server" />

        <ektron:XmlSearchResultsView ControllerID="XmlSearchController1"
          ID="XmlSearchResultsView1" runat="server">
        </ektron:XmlSearchResultsView>

        <ektronUI:Pager ID="pager1" runat="server"
          PageableControlID="XmlSearchController1" ResultsPerPage="10">
        </ektronUI:Pager>
      </div>
    </form>
  </body>
</html>
```

Attributes for XmlSearch

Properties for XmlSearchInputView

- `ControllerID`. The *ID* of the SiteSearchController that provides the data.
- `XmlConfigId`. Long Integer: If content is an XML Smart Form, the ID of its configuration. See also: Copying a Smart Form Configuration.

Properties for XmlSearchController

- `AdvancedQueryText` (Optional). See *Customizing the search behavior with AdvancedQueryText* on page 2083.
- `ID` (Required). Provides the association for the SiteSearchInputView and SiteSearchResultsView server controls. Here is an example of the ID property with the SiteSearchController.

```
<ektron:SiteSearchController ID="SiteSearchController1"
  runat="server" />
```

Properties for XmlSearchResultsView

- `ControllerID`. The *ID* of the XmlSearchController that provides the data.
- `ElapsedTime`. The time required to execute a search.
- `PageInfo`. Display information about search result numbers, such as **Displaying results 11 through 20 of 35**.
  - CurrentPageIndex. Current page.
  - EndCount. Number of results found.
  - NumberOfPages. Number of results divided by results per page.
  - PageCount. Number of the page being viewed currently.
  - ResultCount. Number of the results returned for the search terms.

    IMPORTANT: The result count is only an estimate. The count becomes more accurate you get closer to the final page.

  - ResultsPerPage. Number of results displayed per page.
  - StartCount. 1.
- `Results`. These are fields returned of type SearchResultsData. Display these inside a ListView or Repeater.
  - Date. Last edit date.
  - Summary. Content summary, up to a maximum of 400 characters.
  - Title
  - Url
  - Type: Content, forms, managed assets, and so on.
    For a full list, enter `SiteSearchResultType` into the Object Browser.
- `QueryText`. String that the site visitor entered to begin search.
- `SearchType`. 1 of the following:
  - Advanced
  - Basic
  - None. No search yet.
  - XmlSearch
- `State`. 1 of the following:
  - initialized. Search has not been run yet.
  - no results. No search results were returned.
  - search results. Search results were returned.

Events for XmlSearch

None.

Methods for XmlSearch

- `GetXmlFields(long xmlConfigId);`
  - `GetXmlFields` gets the XML search field list and update the View for the search form.
  - `XmlConfigId` is the Smart Form configuration ID.
  - Return Type: None.
- `XmlSearch(SearchModel model);`
  - Xml Search. `SearchModel` contains all the search results, search query, query filter and paging information of the search.
  - Return Type: None.

Theming for XmlSearch

Theming styles the appearance, including colors and background textures, using jQuery UI CSS Framework and Ektron UI CSS Framework. Use the ThemeRoller tool to create and download custom themes that are easy to build and maintain. If you want to customize it further, you can use control-specific classes referenced within the control's markup.

The classes in the following example are colored and bolded as follows:

- Ektron UI
- jQuery UI
- Control-specific class

```
<div id="uxXmlSearchInput"
  class="ektron-ui-control
  ektron-ui-search
  ektron-ui-search-xml">
  <div class="xml-search-fields">
    <table class="fields-list" id="fields">
      <tbody>
        <tr class="text-field">
          <td class="field-label">
            <label id="uxXmlSearchInput_ctl00_0_aspFields_0_uxField_2_
              ctl00_0_aspLabel_0"
              for="uxXmlSearchInput_ctl00_0_aspFields_0_uxField_2_ctl00_
                0_aspOperators_0">
                CustomerCompany
            </label>
          </td>
          <td>
            <select class="operators"
              id="uxXmlSearchInput_ctl00_0_aspFields_0_uxField_2_ctl00_
                0_aspOperators_0"
              name="uxXmlSearchInput$ctl00$ctl00$aspFields$ctrl2$uxField$
                ctl00$ctl00$aspOperators">
            <option value="Unselected">Unselected</option>
            <option value="Contains">Contains</option>
            <option value="Equals" selected="selected">Equals</option>
            </select>
            <span class="ektron-ui-input     value"
              id="uxXmlSearchInput_ctl00_0_aspFields_0_uxField_2_ctl00_0_
                uxInputText_0_TextField_0_uxTextField_0">
```

```
            <input type="text"
              id="uxXmlSearchInput_ctl00_0_aspFields_0_uxField_2_ctl00_0_
                uxInputText_0_TextField_0_aspInput_0"
              value="aveya"
              name="uxXmlSearchInput$ctl00$ctl00$aspFields$ctrl2$uxField$
                ctl00$ctl00$uxInputText$TextField$aspInput">
            </span>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
  <span class="ektron-ui-button">
  <input type="button" autocomplete="off" value="Search" title=""
    onclick="if (!$ektron.groupValid('urn:Ektron.ValidationGroup.
      uxXmlSearchInput_ctl00_0'))
    {return false;};WebForm_DoPostBackWithOptions
      (new WebForm_PostBackOptions
      ('uxXmlSearchInput$ctl00$ctl00$uxXmlSearchButton', '', true,
      'urn:Ektron.ValidationGroup.uxXmlSearchInput_ctl00_0', '',
        false, true));"
     name="uxXmlSearchInput_ctl00_0_uxXmlSearchButton_0_Button"
     id="uxXmlSearchInput_ctl00_0_uxXmlSearchButton_0_Button"
     class="ui-button ui-widget ui-state-default ui-corner-all"
     role="button">
  </span>
  <script type="text/javascript"
    id="uxXmlSearchInput_ctl00_0_uxXmlSearchButton_0_uxXmlSearchInput_
      ctl00_0_uxXmlSearchButton_0EktronScriptBlockliswr_0">
    <!--//-->
    <![CDATA[//>
    <!--
      Ektron.ready(function(event, eventName)
      {
      $ektron("#uxXmlSearchInput_ctl00_0_uxXmlSearchButton_0_
        Button").button();
      });
    //-->
    <!]]>
  </script>
  <script type="text/javascript"
    id="uxXmlSearchInput_ctl00_0_uxScriptBlockSearch_0_uxXmlSearchInput_
      ctl00_0_uxScriptBlockSearch_0EktronScriptBlockkrmcq_0">
    <!--//-->
    <![CDATA[//>
    <!--
      Ektron.ready(function(event, eventName)
      {
      setTimeout("Ektron.Controls.Search.XmlSearch.init(
        { clientId: 'uxXmlSearchInput_ctl00_0' })", 0);
      });
    //-->
    <!]]>
  </script>
</div>
<div id="uxXmlSearchResults"
  class="ektron-ui-control
```

```
    ektron-ui-search
    ektron-ui-search-results
    ektron-ui-search-results-xml">
  <div class="section no-results">
  </div>
  <ul class="results">
    <li class="result     ektron-ui-clearfix">
      <h3 class="title">
        <a href="http://ws10548b/Clients/Alex-Tishler,-Aveya/">
          Alex Tishler, Aveya</a>
      </h3>
      <div class="summary">  "I have recently purchased
        a copy of OnTrek and am very happy with the software.
        It is amazing how many features are available within
        the network. I am also very impressed by the quality
        and the speed of the technical support."
      </div>
      <span class="url     ektron-ui-quiet">http://ws10548b/Clients/
        Alex-Tishler,-Aveya/</span>
      <span class="date     ektron-ui-quiet">9/7/2011 2:47:07 PM</span>
    </li>
  </ul>
</div>
```

Example for XmlSearch

.aspx

```
<ektron:XmlSearchInputView ID="uxXmlSearchInput" runat="server"
  ControllerID="uxSearchController" XmlConfigId="10" >
</ektron:XmlSearchInputView>
<ektron:XmlSearchResultsView ID="uxXmlSearchResults" runat="server"
  ControllerID="uxSearchController" >
</ektron:XmlSearchResultsView>
<ektron:XmlSearchController ID="uxSearchController" runat="server">
</ektron:XmlSearchController>
```

Creating your own XML search template

You have an understanding of the structure of XML field types, as explained in *Search model for XML Smart Forms* on page 2117.

1. Bind fields to a list view.

```
<ektron:XmlSearchInputView ID="xmlSearchView" runat="server"
  ControllerID="SearchController" XMLConfigId="10">
<ItemTemplate>
  <asp:ListView ID="fields" DataSource='<%# Eval("Fields") %>'
    runat="server">
```

NOTE: The DataSource is the field's array returned from the SearchModel.

2. Insert a set of `<ItemTemplate>` tags.

```
<ektron:XmlSearchInputView ID="xmlSearchView" runat="server"
  ControllerID="SearchController" XMLConfigId="10">
<ItemTemplate>
```

```
  <asp:ListView ID=”fields” DataSource=’<%# Eval(“Fields”) %>’
    runat=”server”> </ItemTemplate>
```

3. Get an XML field and display its corresponding template. Because it does not specify which type of XML Smart Form field is being retrieved, use an XMLSearchField control to determine that.

   The XMLSearchField control has a `Field` property, which maps the control to the correct template in the `\workarea\FrameworkUI\Templates\Search\Fields` folder. For example, a boolean Smart Form field is mapped to `booleanfield.ascx`.

```
<ektron:XmlSearchInputView ID=”xmlSearchView” runat=”server”
  ControllerID=”SearchController” XMLConfigId=”10”>
<ItemTemplate>
  <asp:ListView ID=”fields” DataSource=’<%# Eval(“Fields”) %>’
  runat=”server”>
</ItemTemplate>
<ItemTemplate>
  <ektron:XmlSearchField ID="uxField" runat="server" Field='<%#
    Container.DataItem %>'>
  </ektron:XmlSearchField>
</ItemTemplate>
```

   The XMLSearchField control lets you manage the display of XML Smart Form fields displayed in the ListView. So, you can modify the search input form with effects like the `<ItemSeparatorTemplate>`, which lets you place text, images, and so on, between each field.

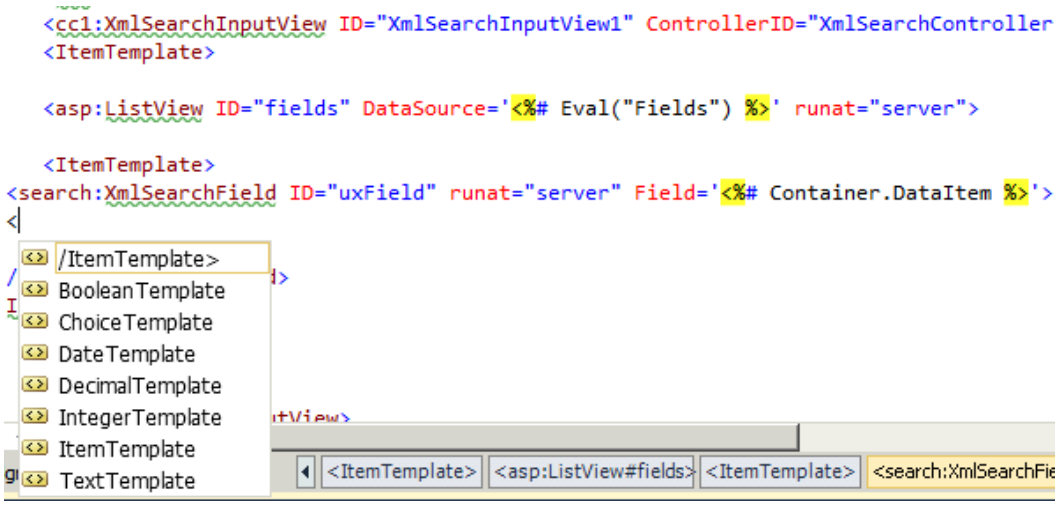
Customizing the rendering of a field

You can customize the behavior of an XML Smart Form field on the input form.

This is a complex procedure because, when a user clicks **Search**, your template must retrieve the modified fields, pass that data to the containing template which, in turn, passes the modified fields to the Search API. Use 2-way data binding to perform this procedure.

You followed the steps in *Creating your own XML search template* on the previous page.

1. Inside your XML Smart Form template, press **Enter** after `Field='<%# Container.DataItem %>'>`.
2. Insert a less than symbol (**<**). A drop-down list of Smart Form field types appears.



3. To replace the value of any field type, insert the corresponding Template field. For example, to replace a text element, click **TextTemplate** to insert a set of `<TextTemplate>` tags. Whatever you enter between the tags replaces the text template on the Smart Form input form.

   For example, to replace a text template field with a text box, use the following syntax.

```
<TextTemplate>
  <asp:TextBox ID="aspValue" Text='<%# Bind("Value") %>'
   runat="server" </asp:TextBox></TextTemplate>
```

Notice that this example uses `Bind`, which is a two-way statement. If the user changes the value in the text box, then posts back, the Bind command pushes the data into the template. The child template then takes the modified data from the template and pushes it back to the data object.

Server Controls

Developers use Microsoft Visual Studio to work with Ektron server controls.

A server control uses API language to interact with the CMS and Framework UI to display the output. You can drag and drop a server control onto an ASPX page to coexist with other components. This includes control for almost everything from content, to user management, breadcrumbs, and social network controls; to provide out-of-the-box markup and functionality. In addition to having a set of properties that you can use to change the controls' output and behavior, you can access the server control API.

Ektron's server controls let you insert many standard methods and properties within the Visual Studio environment. This means that you can see the effect of your changes in real time. You do not have to modify a page then compile a sample project to see the results.

You can insert server controls using drag and drop or programmatically. You can also use databinding to retrieve and display data from Ektron.

Additional information and examples for Ektron server controls are available online after you install the following sample site.

`http://localhost/samplesite/Default.aspx`

Replace *localhost* with the `webroot` where you installed the sample site.

IMPORTANT: For improved security,you should rename or remove the Web services file when you move it to your production server. After installation, this file is named `ServerControlWS.asmx` in the */siteroot/Workarea/* folder in your Web root.

`ServerControlWS.asmx` is the Web service that lets server controls communicate with Ektron. The path is coded in the `web.config` file as follows. Edit this line if you change the location or name of the `ServerControlWS.asmx` file.

```
<!-- Web Service URL for server controls design time -->
<add key="WSPath"
  value="http://localhost/siteroot/Workarea/ServerControlWS.asmx" />
```

The following list shows the Ektron server controls.

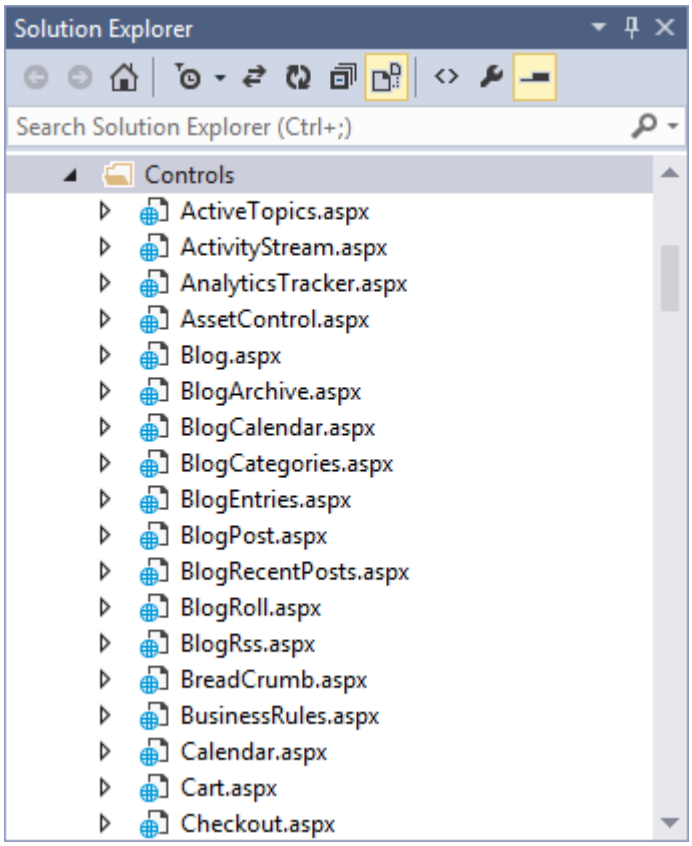
- *ActiveTopics* on page 2165
- *ActivityStream* on page 2219
- *AdaptiveMultiView* on page 1914
- *AnalyticsTracker* on page 2167
- *AssetControl* on page 2170
- *Blog server controls* on page 2173
  - *Blog* on page 2173
  - *BlogArchive* on page 2178
  - *BlogCalendar* on page 2180
  - *BlogCategories* on page 2182
  - *BlogEntries* on page 2184
  - *BlogPost* on page 2188

Opening a sample project in Visual Studio

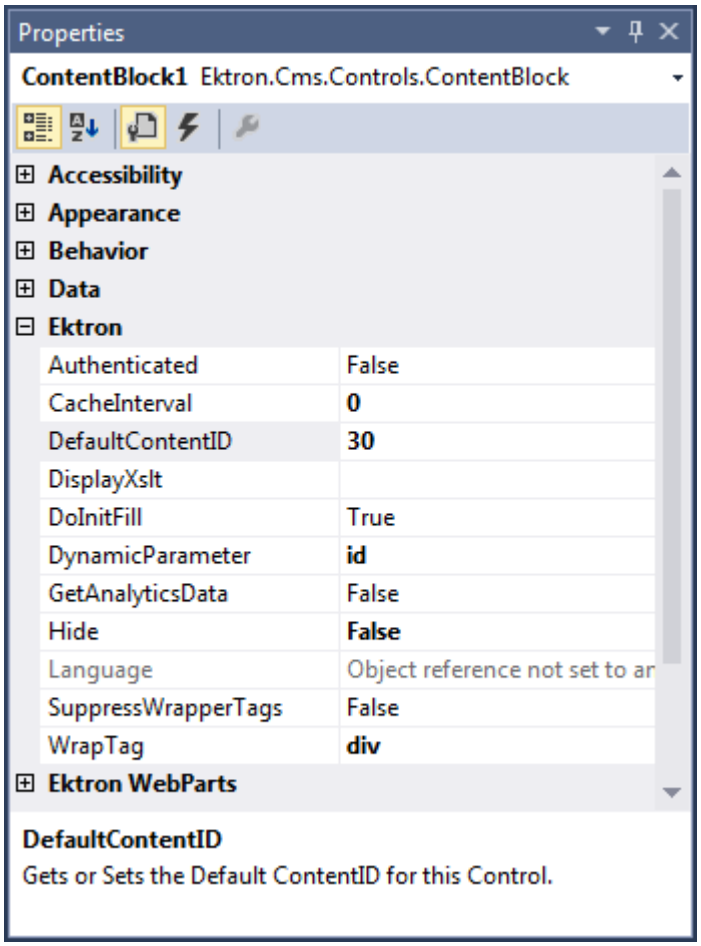
1. Using Visual Studio, choose **File > Open Project...** and browse to and double click Ektron's solution file, `localhost/siteroot/projectname.sln`. For example, for the OnTrek sample site, the file is `OnTrek.sln`. The sample site project opens.

   You can also open your website with **File > Open Web Site...**

2. To work on a template page, open the Solution Explorer and click the page you want.

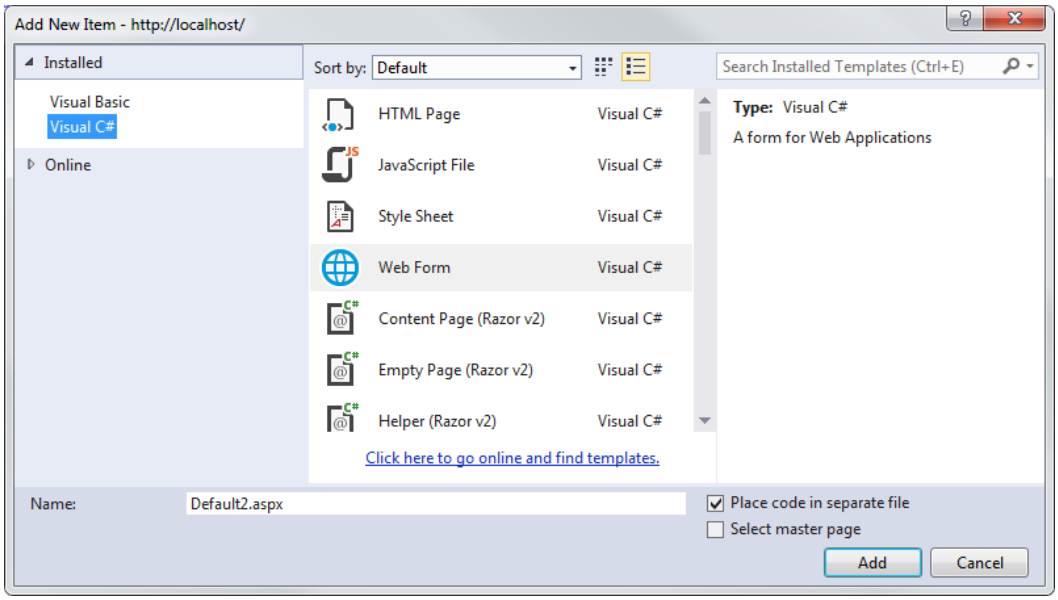


When you select a page, its properties appear in the Properties Window, and the page appears in the center of the screen. A control's properties include several standard .NET properties along with Ektron-specific ones.



Creating a template in Visual Studio

1. Choose **Website > Add New Item...**.
2. On the Add New Item screen, click **Web Form** and assign a name.



3. Add controls to determine the page content.

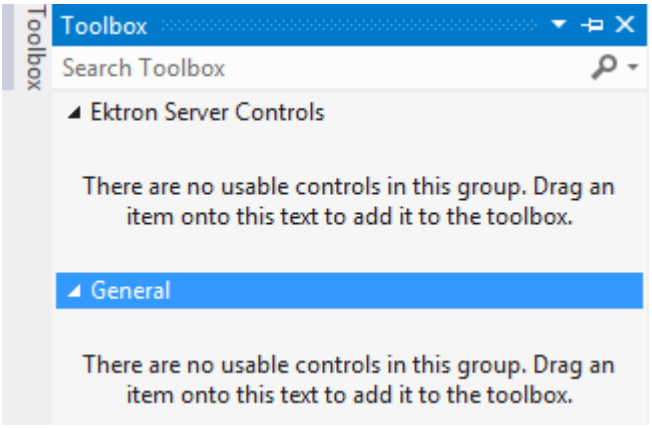
After you open the solution file in Visual Studio and add the required files, build the project. After the project is built, a browser opens and it appears as a Web page. You can also view a Web page while working on it by right clicking on the Web form and clicking **View in Browser**.

Installing server controls into Visual Studio Toolbox

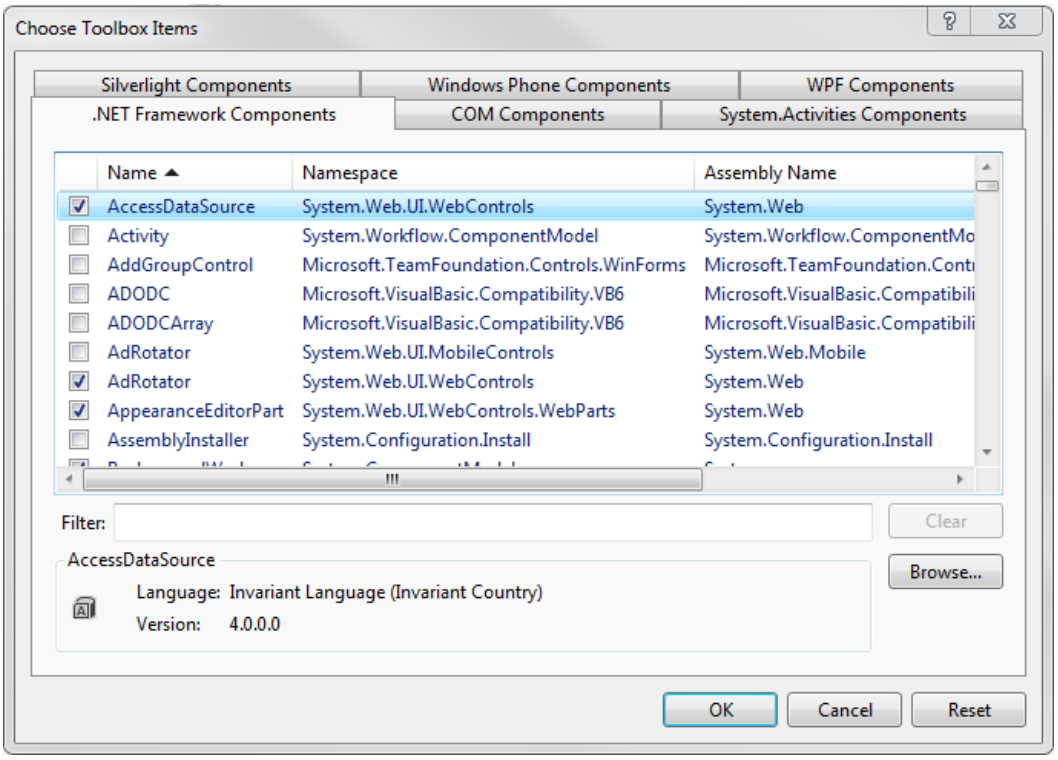
This section describes how to install server controls

NOTE: Search Server controls are installed separately. See *Adding search server controls to Visual Studio* on page 2083.

1. Copy the .dlls to a local drive before installing them. You cannot install them from a network drive.
2. Display the Visual Studio 2010 toolbox (**View > Toolbox**).
3. Right click the mouse within the Toolbox and click **Add Tab**.
4. Type **Ektron server controls** then press **Enter**.

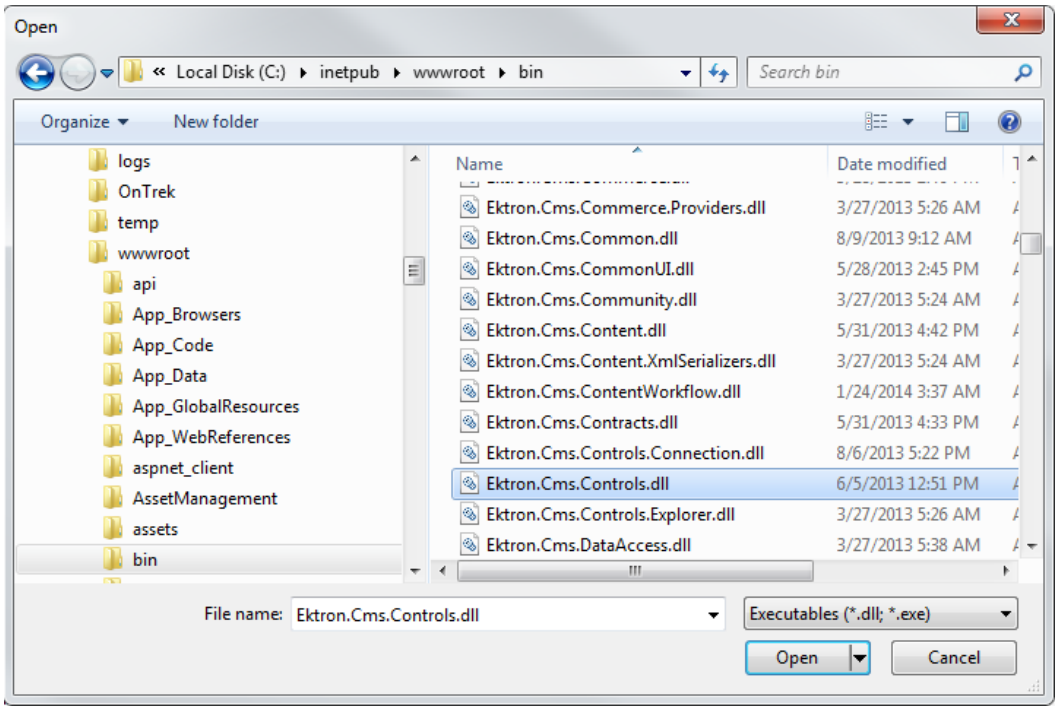


5. Click the Ektron server controls tab to open it.
6. Right click the mouse in the empty area and click **Choose Items**. The Customize Toolbox dialog appears.

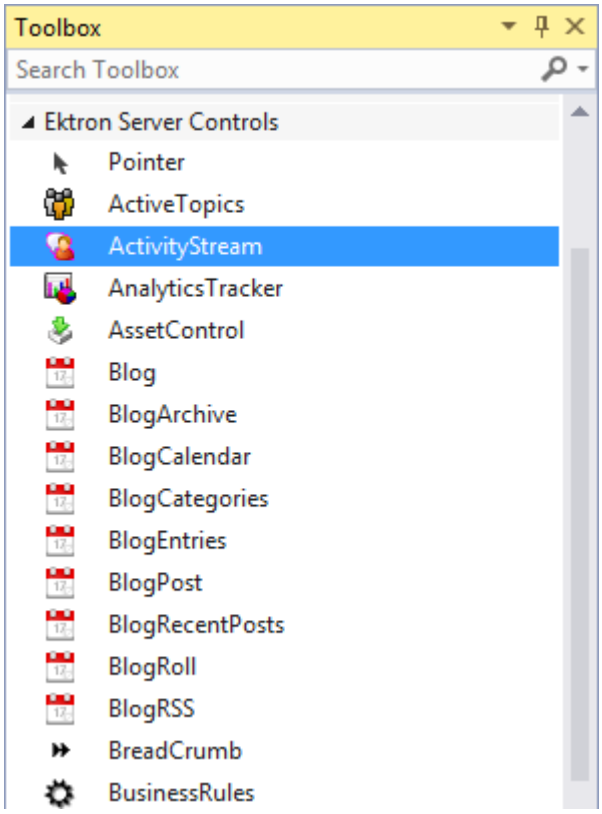


7. Click **Browse...** on the .NET Framework Components tab and find the directory that stores `Ektron`'s dll files (*`siteroot`*`/bin`).
8. Select *`siteroot`*`/bin/Ektron.Cms.Controls.dll` file and click **Open**. This file provides access to Ektron's server controls.

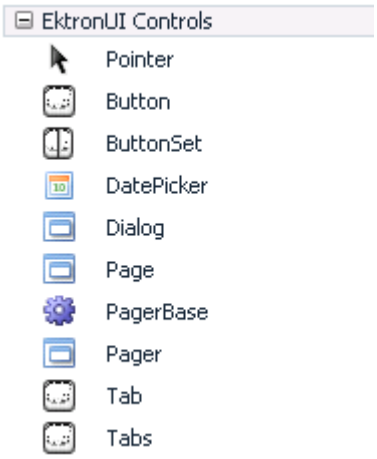
NOTE: Using the bin folder in your site provides better speed when loading Web pages. However, if you use the bin folder located in `C:\Program Files (x86)\CMS400vxx\bin`, you do not have to worry about deleting the .dll file if you change or delete your site. The file is identical in both places.



9. Click **OK** on the Choose Toolbox Items dialog box. The controls are added to the Ektron server controls tab of the Visual Studio Toolbox.



10. To install the UI server controls also, repeat steps 6 through 9 selecting *siteroot*/bin/**Ektron.Cms.Framework.UI.Controls.EktronUI.dll**. The controls appear on the Toolbox tab.



For easier viewing once the server controls are installed, you can right click on them and select **Sort Items Alphabetically**. You can only see the server controls when an ASPX template is selected.

Removing server controls from the Visual Studio Toolbox

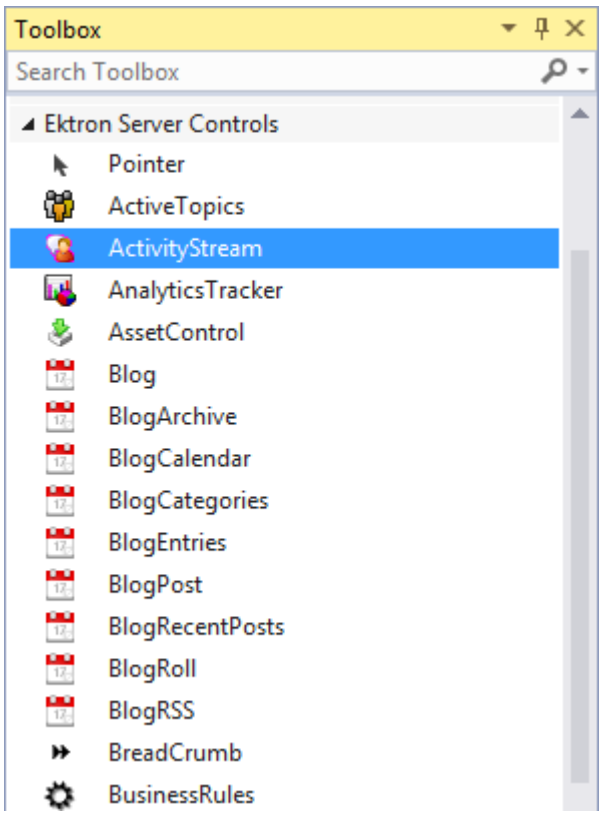
1. Display the Visual Studio toolbox (**View > Toolbox**).
2. Right click the mouse within the Toolbox.
3. Click **Choose Items...**.
4. Click **Namespace** or **Assembly Name** to sort the server controls by manufacturer.
5. Uncheck boxes next to the server controls you want to remove.
6. Click **OK**.

NOTE: If you want to delete all of the server controls from the **Ektron server control** tab, right click on the tab and choose **Delete Tab**.

Inserting a server control using drag and drop

Because Visual Studio is a visual environment, you can watch the page layout change as you add or move a control and adjust its properties. Use the CMS Explorer whenever you need to identify an Ektron object (such as content block or collection).

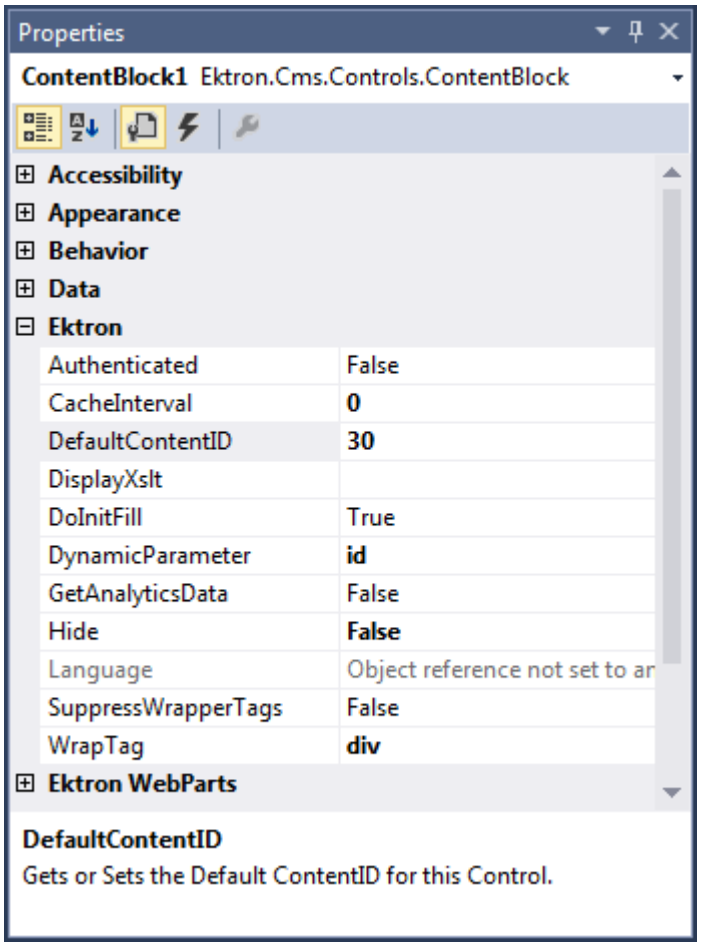
1. Display the Visual Studio toolbox (**View > Toolbox**).
2. Click the **Ektron server control** tab to display the server controls.



3. Drag a server control and drop it into the desired location on the page.

> **NOTE:** You also can place the cursor on the page where you want the server control, then double click the server control that you want.

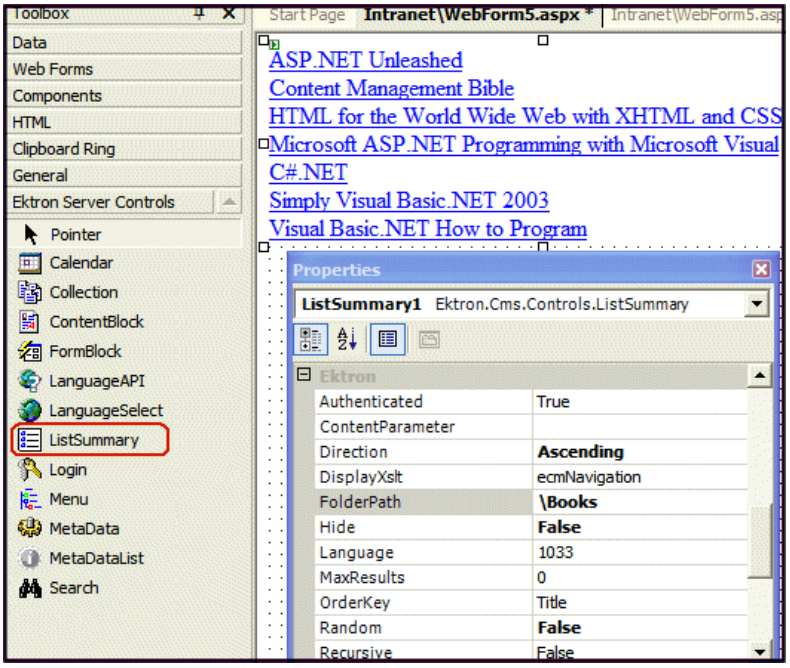
4. Modify the control's properties using the properties window of Visual Studio. The following images shows the ContentBlock properties.



Modifying a server control after drag and drop

You can manipulate a server control after dragging and dropping it on a Web form by using the code-behind. The following example shows using a drag and drop ListSummary server control then modifying it programmatically.

1. Drag and drop a ListSummary on your Web form and set your properties.



2. Add the following code to the code-behind.

```
Dim myString As String
Dim i
For i = LBound(ListSummary1.EkItems)To UBound(ListSummary1.EkItems)
  myString &= "<a href=""" & ListSummary1.EkItems(i).QuickLink & """>"
  & ListSummary1.EkItems(i).DateCreated & "</a><br>"
ListSummary1.Text = myString
Next
```

ListSummary1 is the ID of the object. It is used to get access to its properties.

3. Create a string that contains the output (myString).

```
Dim myString As String
```

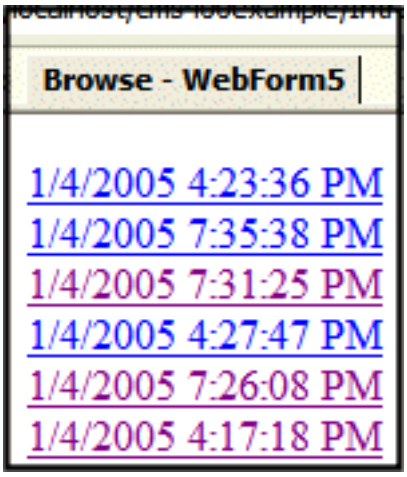
4. Set the object's `Text property to that string.`

```
myString &= "<a href=""" & ListSummary1.EkItems(i).QuickLink & """>"
  & ListSummary1.EkItems(i).DateCreated & "</a><br>"
```

5. Wrap it in a loop so it loops through each of the items.

```
Dim i
For i = LBound(ListSummary1.EkItems) To UBound(ListSummary1.EkItems)
Next
```

This example outputs the date created for each content block in a ListSummary.



Inserting a server control programmatically

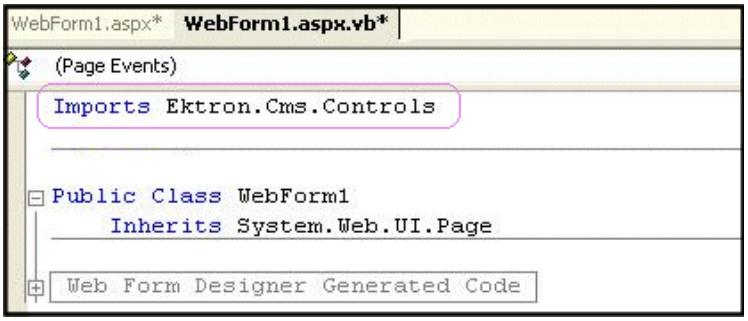
You may want to insert server controls programmatically for the following reasons:

- You want the control to be loaded into memory only under certain conditions. In this case, insert the logic that only displays the control if the condition exists.
- You want to display only certain properties of an object, such as the title of last edited date of a content block.

To insert an Ektron server control programmatically:

1. Declare the namespace at the top of the code-behind Visual Basic file.

   NOTE: You do not need to declare a namespace. However if you do not, you must fully qualify objects that you create. For any customization of Ektron, classes or controls that inherit from Ektron classes, you should create your own namespace within `Ektron.Cms.Custom`. For example, if your company is AcmeExampleTech, Inc. You should create all of your custom classes within the namespace `Ektron.Cms.Custom.AcmeExampleTech`.

2. Create an instance of the new control by declaring a control as an object in the code-behind; in this example, a collection named `MyColl`.

   ```
   Ektron.Cms.Controls.Collection MyColl =
     new Ektron.Cms.Controls.Collection();
   ```

   You can declare any server control as an object by using the server control name. Another example would be: `Dim MyMdl as New MetaDataList`.

3. Set the properties that you want to display on the page. For example:

```
Dim MyColl as New Collection
MyColl.DefaultCollectionID = 4
MyColl.Page = Page
```

With C#, use this syntax.

```
>
Ektron.Cms.Controls.Collection MyColl =
  new Ektron.Cms.Controls.Collection();
MyColl.DefaultCollectionID = 4;
MyColl.ID = "Collection1";
MyColl.Page = Page;
```

These lines tell the page to display CollectionID 1 unless otherwise specified.

IMPORTANT: When using code-behind to add a server control to your Web form, you must set the Page object for the server control to Page. For example, `Mycoll.Page = Page`
This line needs to appear between Dim new server control line and the Fill() line. This line is not added when dragging and dropping a server control on a Web form.

If you do not know an object's ID number, you can switch to Design mode, drag and drop the object, then use the CMS Explorer to find the ID number. If you do this, remember to delete the dropped object when you are done. You can also obtain the ID number via the Workarea. This line sets the `Random` property to **true**.

```
Dim MyColl as New Collection
MyColl.ID = "Collection1"
MyColl.DefaultCollectionID = 4
MyColl.Page = Page
MyColl.Random = True
```

4. Call the Fill method to fill an object's properties on the page because there is no render event when using objects as components not as controls.

```
>
Dim MyColl as New Collection
MyColl.ID = "Collection1"
MyColl.DefaultCollectionID = 4
MyColl.Page = Page
MyColl.Random = True
MyColl.Fill()
```

5. Use the property to determine what appears on the Web page. For example, to display the first item in a collection, use this syntax.

NOTE: Before adding this line you need to drag and drop a label on your Web form.

```
>
Dim MyColl as New Collection
MyColl.ID = "Collection1"
MyColl.DefaultCollectionID = 4
MyColl.Page = Page
MyColl.Random = True
MyColl.Fill()
Label1.Text = myColl.EkItems(0).Title
```

To display *all* items in a collection, use this syntax.

```
Dim myColl As New Ektron.Cms.Controls.Collection
Dim ekitem As New Ektron.Cms.Common.ContentBase
MyColl.DefaultCollectionID = 2
MyColl.ID = "Collection1"
MyColl.Page = Page
MyColl.Fill()
Label1.Text = "<ul>"
For Each ekitem In myColl.EkItems
Label1.Text &= "<li><a href=""" & ekitem.QuickLink & """>"
  & ekitem.Title & "</a>"
Next
Label1.Text &= "</ul>"
```

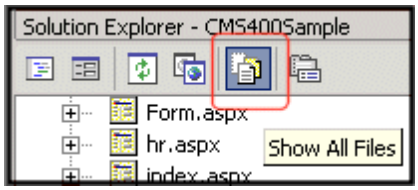
This example displays the quick link for every content block in the collection, formatted as a bulleted list. You can use similar code to display a List Summary or search results. The following explains the new (red) code above.

- `label1.Text = "<ul>"` displays the opening tag for the bulleted list
- `For Each ekitem In myColl.EkItems` creates a loop for all content blocks in the collection
- `label1.Text &= "<li><a href=" & ekitem.QuickLink & ">" & ekitem.Title & "</a></li>"` for each content block in the collection, displays its quicklink and title
- `Next` loops through all content blocks in the collection
- `label1.Text &= "</ul>"` closes the bulleted list

Customizing a server control

Visual Studio separates coding and logic from presentation. Web page formatting is handled by a page's HTML, while the logic is handled by the code-behind, which is stored in the corresponding .vb file. For example, if the ASP.NET page is `mypage.aspx`, the code-behind file is `mypage.aspx.vb`.

NOTE: If you do not see the code-behind files, click **Show All Files** on the Visual Studio Solution Explorer toolbar (circled in the following figure).



Within the Visual Basic file, you can use Visual Basic to insert code to manipulate the events that occur on the page.

This subsection contains the following topics:

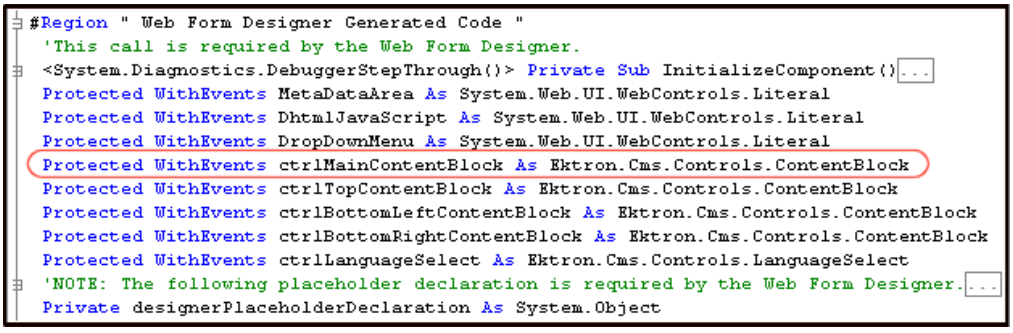
Viewing a server control in HTML

Within a Web page's HTML, a <cms> tag wraps the Visual Studio object. The following is an example of the Search and Contentblock server controls.

```
<cms:Search id="Search1" runat="server" ButtonText="Search"
  Display="Vertical">
</cms:Search>
<cms:ContentBlock
  id="ctrlMainContentBlock"
  runat="server"
  DefaultContentID="1"
  DynamicParameter="id"
  OverrideXslt="Default">
</cms:ContentBlock>
```

Viewing a server control in the code-behind

Within a Visual Studio code-behind file, the Ektron server controls appear (along with the Visual Studio controls) in the Web Form Designer Generated Code section. When you click **+** to display this section, you see something like the following. The content block listed in HTML (from the previous section) is circled to help you see their relationship.



The next section of the code-behind page loads the page into the browser.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
  System.EventArgs) Handles MyBase.Load
```

You want your events to occur while the page is loading, so add the custom code following this line.

Customizing a server control in the code-behind

To customize an Ektron server control in the code-behind, insert code similar to the following *after* the Page_load command.

```
Dim MyObj As New Ektron.Cms.Controls.ContentBlock
```

This code declares a variable named `MyObj` and assigns to it the value of a content block. The content block is part of the Ektron.CMS.Controls content base, so it has access to the Ektron database.

After defining MyObj as a content block, you can access its properties. For example, to assign a defaultID of 24, insert the following.

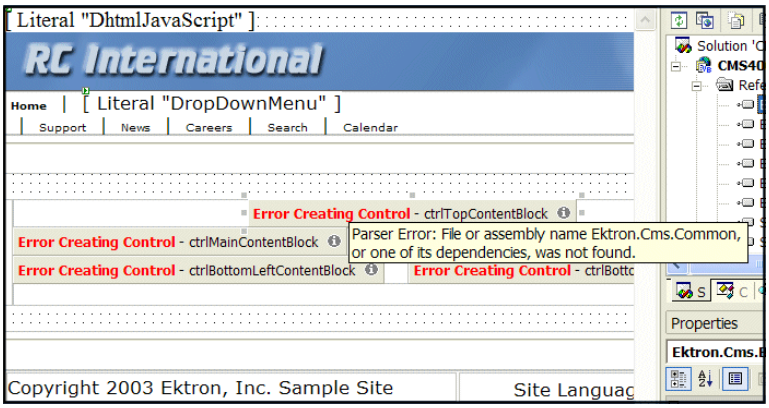
```
Dim MyObj As New Ektron.Cms.Controls.ContentBlock
MyObj.DefaultContentID = 24
```

The content block can be specified dynamically in the URL of the hyperlink that calls it. If not, content block 24 displays.

This is an example of programmatically applying property values to content blocks.

Troubleshooting error creating control message

If you get an Error Creating Control message while trying to use a server control, you can view the message by hovering the mouse over the control.



Using Ajax-enabled server controls and custom code

BEST PRACTICE

> When using Ajax server controls and custom code, wrap the custom code in a check for "is not a callback," so it is not executed when a callback from an Ajax server control occurs.

If you use an Ajax-enabled server control and write custom code, it may generate an exception, indicating the server control does not work. This issue happens during callback for the Ajax server control when custom code accesses a property that is filled during page load, but not filled during callback.

For example, a Poll server control is on a Web form, and you want the title of a content block to appear in a literal on that form. The `EkItem.Title` property for the content block is filled upon page load. When a site user answers a poll question, an exception occurs during the callback because the `EkItem.Title` property is not refilled. However, the site user does not see the exception because it looks like the Poll server control is not working.

The following example shows custom code that makes the content block's title appear in the literal:

```
Literal1.Text = ContentBlock1.EkItem.Title
```

To solve this issue, wrap the custom code in a check for "is not a callback". This prevents the code execution when callback occurs. For example:

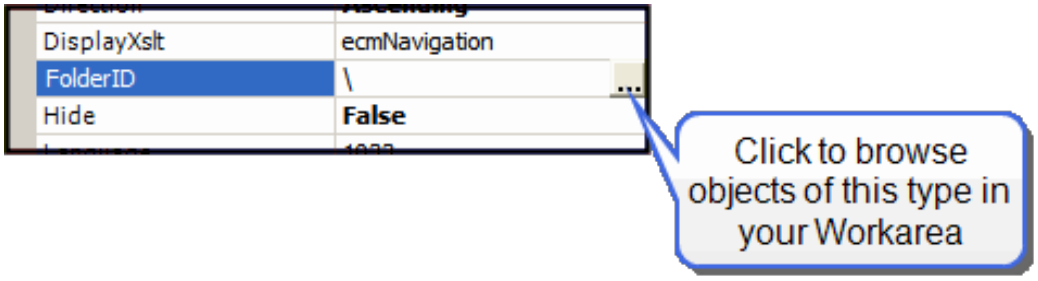
```
[C#]
If( !IsCallback )
  {
    Literal1.Text = ContentBlock1.EkItem.Title
  }

[VB]
If
  ( Not IsCallback )
  Literal1.Text = ContentBlock1.EkItem.Title
End If
```

Browsing your Ektron site using CMS Explorer

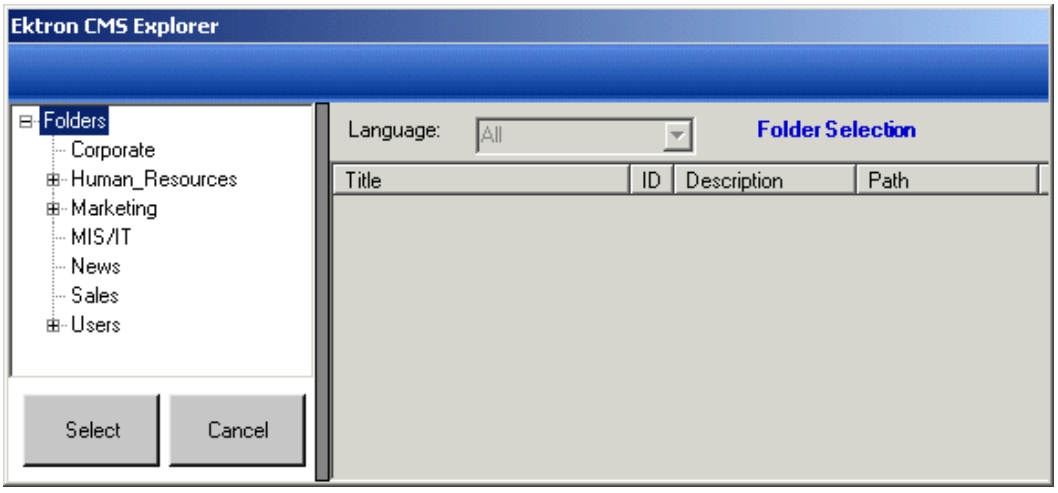
CMS Explorer lets you browse your website to identify Ektron objects such as folders, calendars, blogs, and content blocks. You access the CMS Explorer from the Properties window. For example, if you insert a *ListSummary* on page 2451 server control, its `FolderID` property identifies the folder whose contents are displayed.

1. If you are not sure of the folder path, click the ellipsis (**...**) button next to the property field. A login screen appears.

   NOTE: You cannot be logged into the CMS Explorer and the Workarea at the same time. If you log into the CMS Explorer while logged into the Workarea, an error message appears. If you log into the Workarea while logged into the CMS Explorer, you will need to re-login to the CMS Explorer when you return to using it.



2. Enter your Ektron username and password. The CMS Explorer window appears so you can browse your Ektron website.



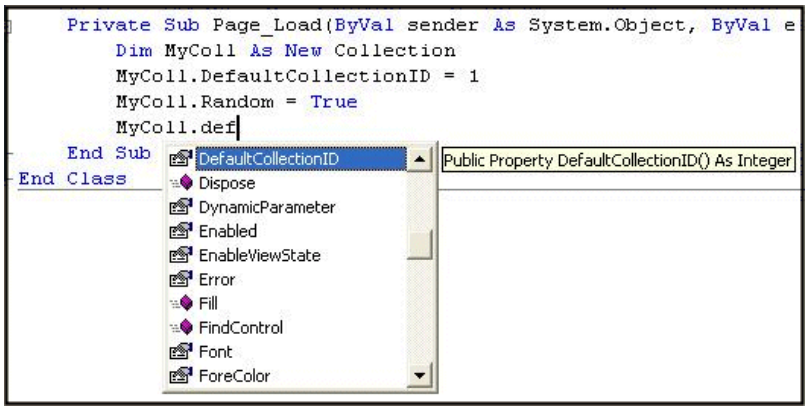
3. Navigate through the folders, select an object, then click **Select**. The selected object is pasted to the server control property.

NOTE: Although you see the object in the selected language in Visual Studio, the language is not stored. For example, if you select a German content block whose ID=2, Ektron only stores `content block ID=2`. When a visitor to your site browses to that page, the content block does not appear in the selected language. Instead, it is determined by a cookie or the user's language selection.

Accessing server control properties in code-behind programmatically

Every server control has properties associated with it that you can access only programmatically. This section explains what they are, how to access them, and how to use them.

You can use Visual Studio's IntelliSense feature to display a control's properties. The IntelliSense box appears as soon as you insert the period (.) after the object, as illustrated below.



The IntelliSense box displays *all* properties that can be applied.

The property's tooltip text indicates its type. In the previous example, note that the `DefaultCollectionID`'s type is integer.

To learn about native Visual Studio properties, see the Visual Studio documentation. For more information about accessing Ektron object properties, see *Customizing a server control in the code-behind* on page 2145.

Personalizing a page with user names and IDs

Every server control has the following read-only properties in the code-behind that let you personalize any page with user names and IDs, and show if they are logged in.

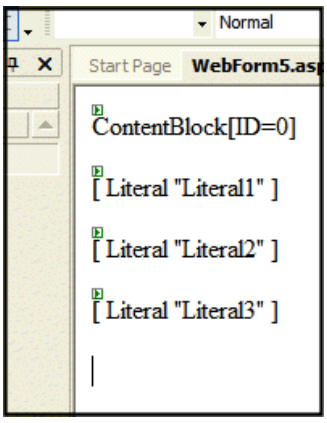
NOTE: The properties do not display values within Visual Studio during design time. Instead, they only display values at run time, which are dependent on the user's login status.

- **IsLoggedIn** (Boolean)
  Tells if a user is logged in to Ektron.
  - **True**. User is logged in
  - **False**. User is not logged in
- **LoggedInUserName** (String)
  Gets the Ektron user name to display.
- **LoggedInUserID** (Integer)
  Gets the Ektron ID of the user to display.

The following example shows these properties in code-behind.

NOTE: You must be logged in to Ektron for this example to show your name and ID.

1. Drag an Ektron server control onto a Web form.
2. Drag 3 Literals onto the Web form.

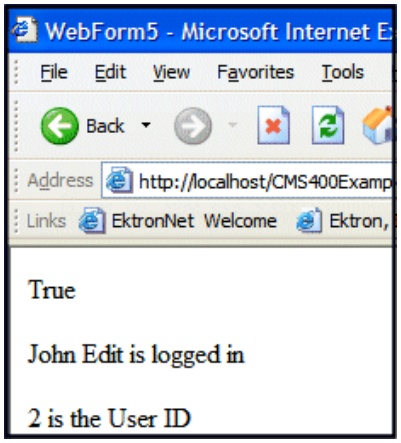


3. Open the code-behind for the Web form.
4. Add the following code to the Page_Load event.

```
Literal1.Text = ContentBlock1.IsLoggedIn
If ContentBlock1.IsLoggedIn Then
```

```
  Literal2.Text = ContentBlock1.loggedInUserName
  & " is logged in " Literal3.Text = ContentBlock1.loggedInUserID
  & " is the User ID "
End If
```

5. Build and run the solution.
6. Browse to the login page and log in.
7. Browse to the new Web form you added. The login information appears.

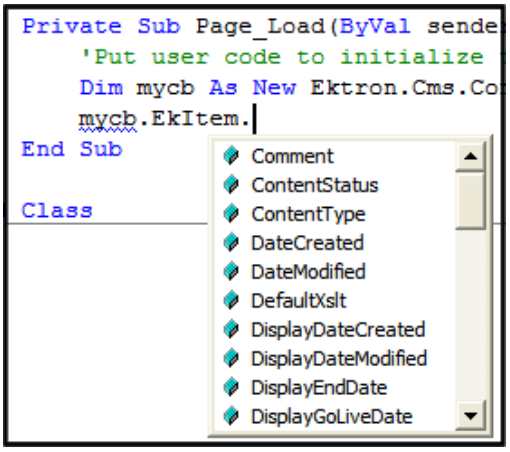


Accessing additional properties

Ektron provides access to additional properties for the following objects.

- ListSummary
- collection
- Search
- ContentBlock
- FormBlock

You can use IntelliSense to select from a list of additional object properties.



To access additional properties, use the standard property's syntax but add `.ekitem` or `.ekitems` after the object.

- *EkItem* is a single `Ektron.Cms.Common.ContentBase`
- *EkItems* is an array of `Ektron.Cms.Common.ContentBase`

```
dim MyCB as New ContentBlock
MyCB.DefaultContentID = 30
MyCB.ID = "ContentBlock1"
MyCB.Page = Page
MyCB.Fill()
label1.text = MyCB.EkItem.dateCreated
```

or

```
dim MyCB as new Ektron.Cms.Controls.ContentBlock
MyCB.DefaultContentID = 30
MyCB.ID = "ContentBlock1"
MyCB.Page = Page
MyCB.Fill()
label1.text = MyCB.EkItem.dateCreated
```

IMPORTANT: To access additional properties for the Collection, ListSummary, and Search objects, use ekitems, not ekitem. For example: `MyColl.ekitems(0).dateCreated`. where (0) is the index of the array. See also: *Accessing items in an array* on page 2153.

The additional properties are as follows.

NOTE: The following properties are read-only. For example, you can get a content block's ID and pass it to another part of the code. However, you cannot set a content block ID to be shown.
For example, `mycb.Ekitem.id = 8` does not set a content block's ID. To set a content block's ID, use `DefaultContentID = 8`.

- **Comment**. The content block's comment.
- **ContentStatus**. The status of the content block.
  - approved
  - checked out
  - checked in
  - expired
  - pending deletion
  - pending expiration
  - pending start date
  - submitted
- **ContentType**. One of the following:
  - all types
  - archived content
  - archived forms
  - content
  - forms
- **DateCreated**. The date when the content block was created, formatted as a .NET date type.

- **DateModified**. The date when the content block was modified, formatted as a .NET date type.
- **DefaultXslt**. The default XSLT used to display the content.
- **DisplayDateCreated**. The date when the content block was created. It is formatted as a string that represents Ektron's display of the date.
- **DisplayDateModified**. The date when the content block was edited. It is formatted as a string that represents Ektron's display of the date.
- **DisplayEndDate**. The content block's end date. It is formatted as a string that represents Ektron's display of the date.
- **DisplayGoLiveDate**. The content block's start date. It is formatted as a string that represents Ektron's display of the date.
- **DisplayStartDate**. The content block's start date. It is formatted as a string that represents Ektron's display of the date.
- **EndDate**. The content block's end date, formatted as a .NET date type.
- **EndDateAction**. One of the following:
  - archive display
  - archive expire
  - refresh report
- **FolderID**. The ID of the folder that contains each content block.
- **GoLiveDate**. The content block's start date formatted as a .NET date type.
- **Html**. The content that makes up the content block. If content block is in XML it will return it as raw XML content.
- **Hyperlink**. Content block title wrapped by `<a href>` tags.
- **Id**. The content block 's ID number.
- **InheritedFrom**. If folder permissions are inherited, the folder from which they are inherited.
- **IsInherited**. Whether a content block's permissions are inherited.
- **IsPrivate**. Whether or not a content block is private.
- **Language**. The content block's language.
- **LastEditorFname**. The first name of the last person to edit the content block.
- **LastEditorLname**. The last name of the last person to edit the content block.
- **PackageDisplayXSLT**. If the content block is XML, the name of its XSLT.
- **QuickLink**. The content block's quicklink.
- **StartDate**. The content block's start date formatted as a .NET date type.
- **Status**. The status of the content block.
  - approved
  - checked out
  - checked in
  - expired
  - pending deletion
  - pending expiration

    - pending start date
    - submitted
- **Teaser**. The content block summary.
- **TemplateLink**. Currently empty and not being used with the ContentBlock server control.
- **Title**. The content block title.
- **UserID**. Last user who edited the content.
- **Xslt1**. The content block's first Xslt, as defined in Ektron.
- **Xslt2**. The content block's second Xslt, as defined in Ektron.
- **Xslt3**. The content block's third Xslt, as defined in Ektron.
- **Xslt4**. The developer can use this property programmatically. Ektron only uses Xslt1, 2 and 3 in the workarea.
- **Xslt5**. The developer can use this property programmatically. Ektron only uses Xslt1, 2 and 3 in the workarea.

Accessing items in an array

To access and manipulate content blocks returned by an object, use the common class `Ektron.Cms.Common.ContentBase`. For example, Search, Collection and ListSummary have `EkItems` (an array of ContentBase), while ContentBlock has a single `EkItem`. This example creates a bulleted list of every item in the collection.

```
dim MyC as new Ektron.Cms.Controls.Collection
MyC.DefaultCollectionID = 1
MyC.ID = "Collection1"
MyC.Page = Page
MyC.Fill()
dim item as Ektron.Cms.Common.ContentBase
MyC.Text = "<ul>"
for each item in MyC.EkItems
MyC.Text &= "<li>" & item.Title & "</li>"
next
MyC.Text &= "</ul>"
Response.Write(MyC.Text())
```

NOTE: For information on using `ekitems` with eCommerce server controls, see Customizing eCommerce server controls with Event Hooks.

Referencing a parent page

Server controls require a reference to their parent page (for example, using the `DynamicParameter` property on a content block to check for a query string). You *must* provide access to the page object if you declared your control in the code-behind. To do this, set the control's `Page` property to the Web page you're working on, as shown in the following example.

```
Ektron.Cms.Controls.Search MySearch = new Ektron.Cms.Controls.Search();
MySearch.Page = Page;
MySearch.Fill();
```

This relationship only is required when inserting a control in the code-behind. When dragging and dropping, even if you make changes in code-behind, the relationship is automatically generated.

BEST PRACTICE

You should include the `Page` property reference when using server controls as components in the code-behind.

Data binding with server controls

With data binding, you can bind Ektron server controls to a GridView Control, DataList Control, or Repeater Control. This gives more flexibility when you use data from the Ektron server controls. Benefits of data binding include ease of data manipulation and the ability to format data.

WARNING! When HTML is bound to a column, you need to add `HtmlEncode = False` to it. Otherwise, the HTML appears as code. For example, `<p>Ektron Inc., an innovator in Web content management software, today announced...</p>`.

The following are data bindable Ektron server controls.

- Collection
- ContentList
- IndexSearch
- ListSummary
- MetaDataList
- RssAggregator

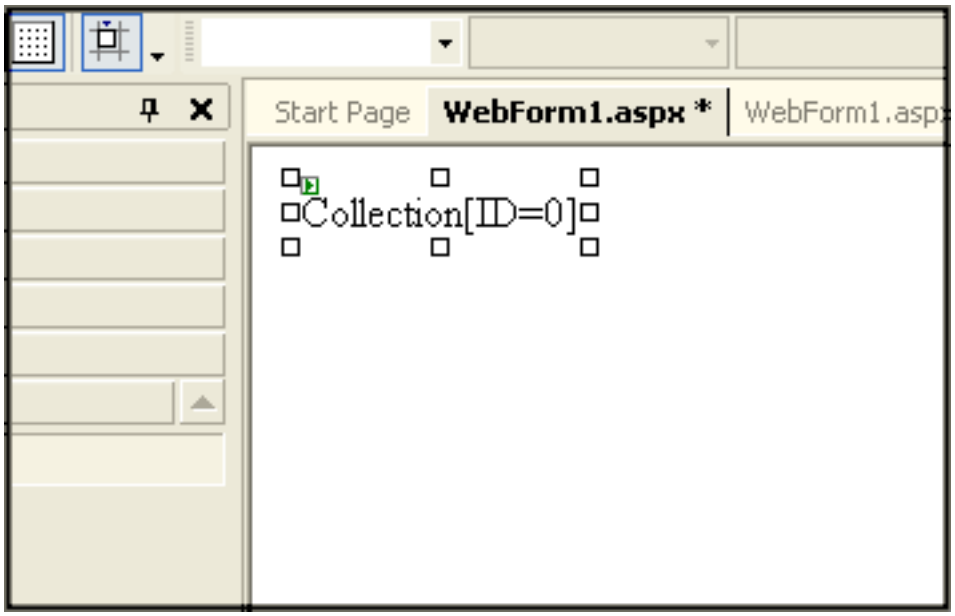
The following example shows code-behind that uses a GridView to display a Collection:

NOTE: For the example code to work properly, you need to drag and drop a GridView server control on a Web form.

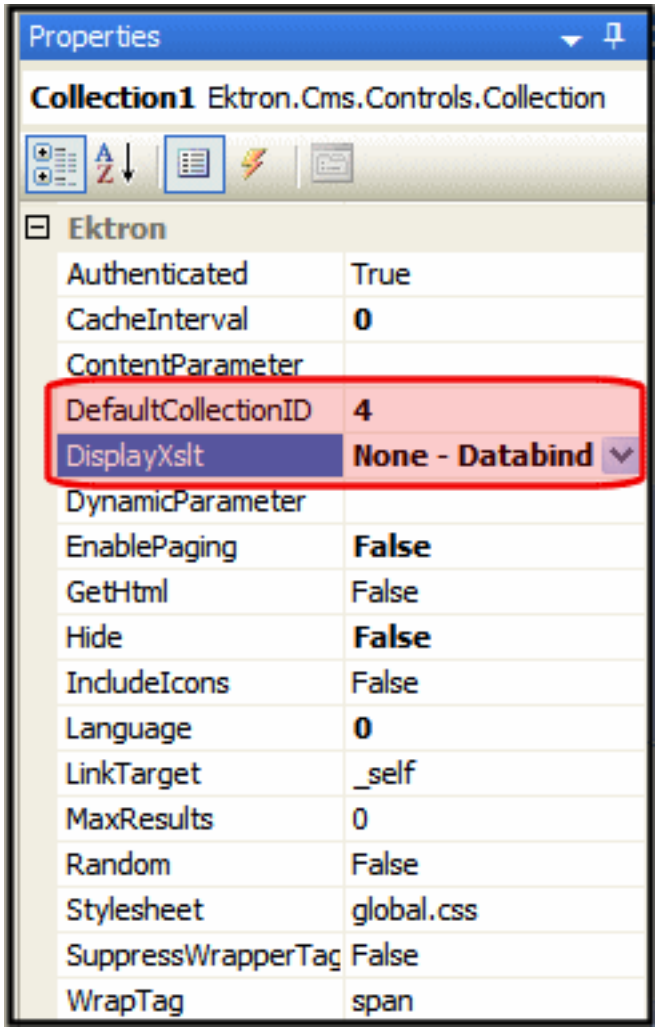
```
Dim myCol As New Ektron.Cms.Controls.Collection
  MyCol.ID = "Collection1"'''Create an ID for the Collection
myCol.DefaultCollectionID = 4
  myCol.Page = Page
myCol.Fill()
GridView1.DataSource = myCol
GridView1.DataBind()
```

Steps to data binding using drag and drop server controls are as follows:

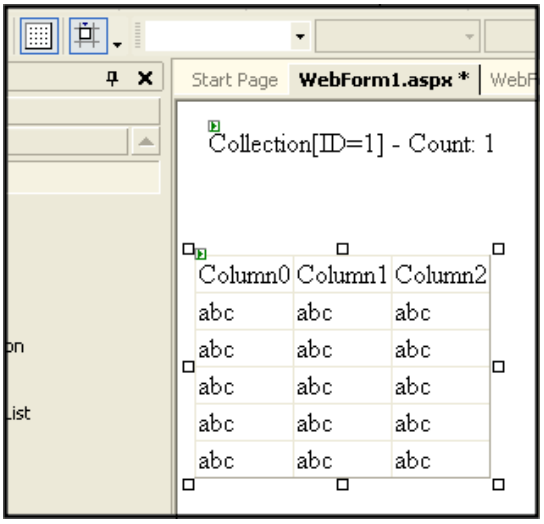
1. Create a new Web form.
2. Drag and drop a data bindable server control on the Web form. For example, a Collection server control.



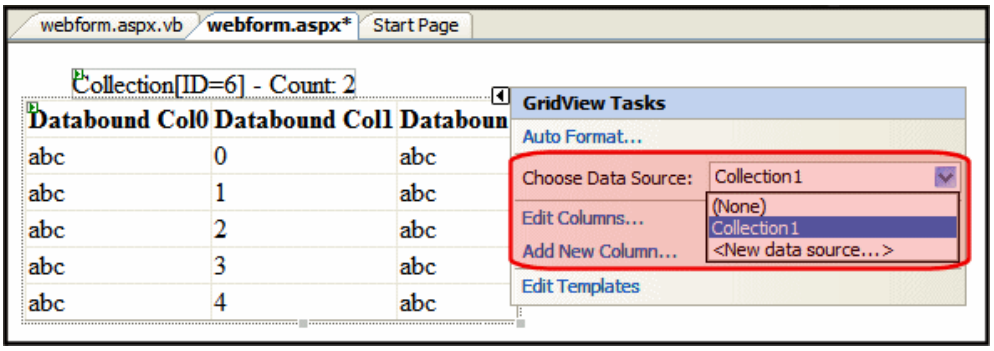
3. In properties, choose a `DefaultCollectionID` for the server control and make sure DisplayXslt is set to **None- DataBind Only**.



4. Drag and drop a GridView on the Web form.



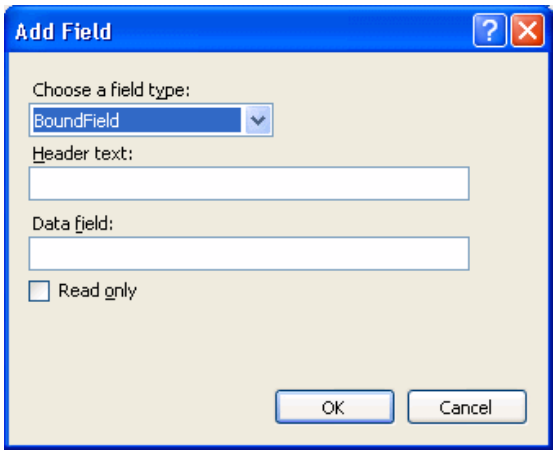
5. In the GridView Tasks, choose the **DataSourceID**.



6. In code-behind, add the following line of code to the Page add field.gif unit event:

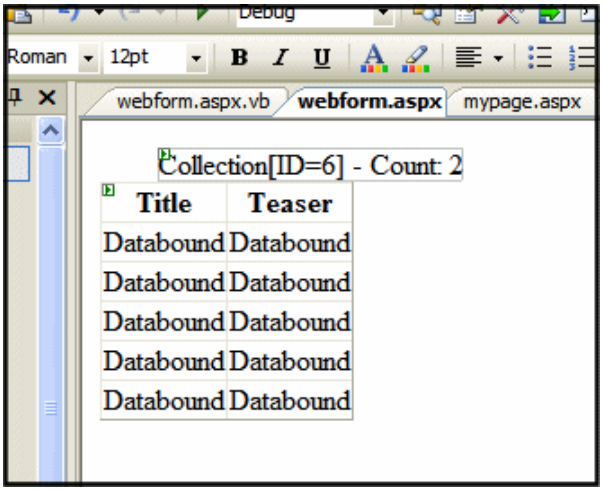
```
Collection1.Fill()
```

7. Select the columns to be databound by clicking **Add New Column** in GridView Tasks.
8. Complete the Add Field window.

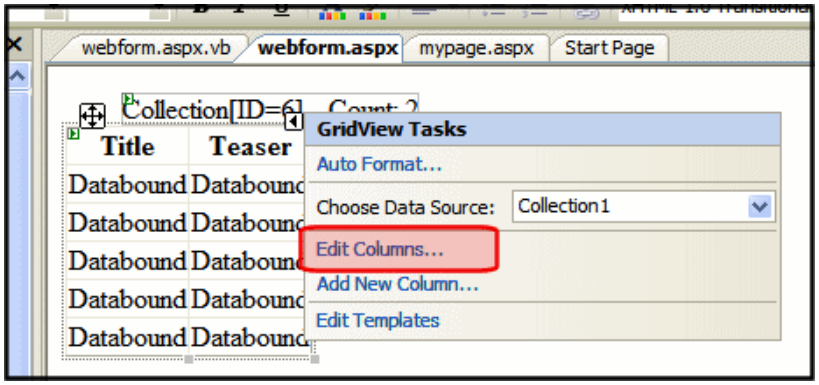


- **Choose a field type**. BoundField
- **Header text**. The title name for each column.
- **Data field**. The data to bind to each column. Choose one of the following:
  - **ID**. The content block ID
  - **Title**. The content block title
  - **Teaser**. The content summary
  - **Html**. The HTML content
  - **StartDate**. The content's start date
  - **DateModified**. The content's last modified date
  - **EndDate**. The content's end date
  - **LastEditorFname**. The last editor's first name
  - **LastEditorLname**. The last editor's last name
  - **QuickLink**. The content's quicklink
  - **HyperLink**. The content's hyperlink
  - **DisplayStartDate**. The string representation of the start date
  - **FolderID**. The folder ID where the content is located
  - **ContentStatus**. The content's status
  - **Language**. The content's default language
  - **DisplayDateModified**. String representation of the content's last modified date
  - **DisplayEndDate**. String representation of the content's end date
  - **EndDateAction**. An action tied to end date. For example, Refresh_Report
  - **Comment**. The content's comments

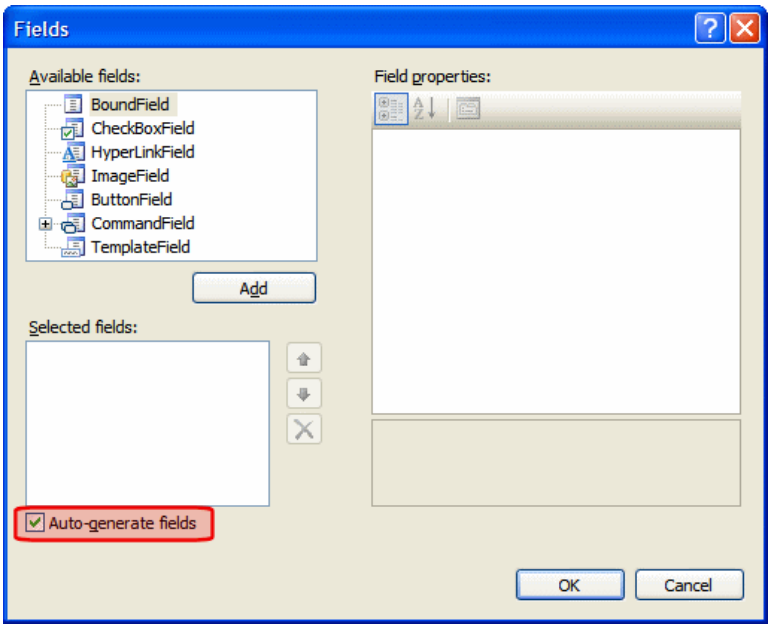
9. Click **OK**.
10. Repeat steps 6, 7 and 8 for each column you want to add.



11. If you want to add all of the columns automatically, in the GridView Task menu click **Edit Columns**. Otherwise, skip to step 13.



12. Click the **Auto-generate fields** checkbox.



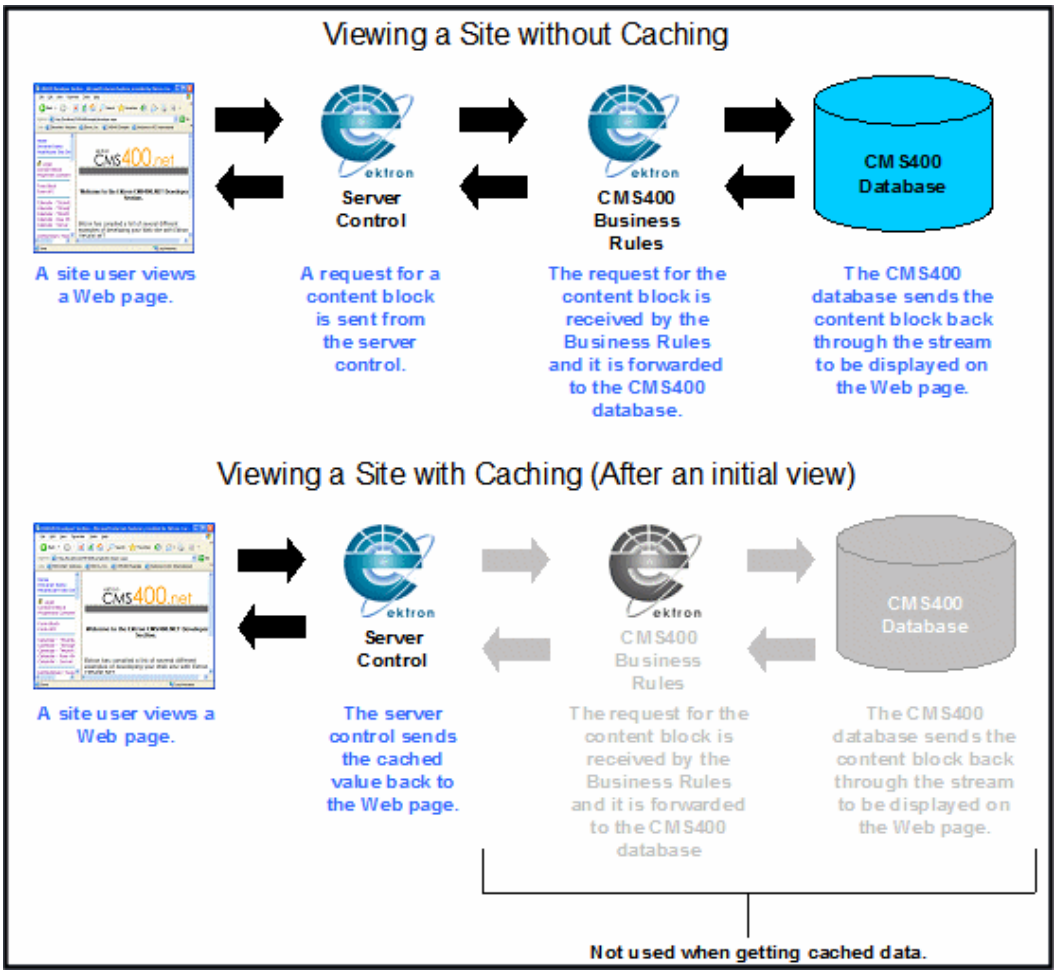
13. Click **OK**.
14. From the Build menu, click **Build Page**.
15. In design view, right click the form and select **View in Browser**.

Title	Teaser
Ektron Rated Positive	Ektron Inc., an innovator in Web content management software, today announced ...
Ektron to Demonstrate Healthcare	"Healthcare Content Indexing Framework" which enables hospitals, medical groups and health insurers to deliver indexed content via the Web.

For more information on GridView, DataList, Repeater and DetailsView, see the help inside Visual Studio.

Caching with server controls

High-performance, scalable Web applications store items in memory after the first time they are requested. The items include data objects, pages, and parts of a page. Caching saves and later reuses page output or application data across HTTP requests. You can store items on the Web server or other software in the request stream, such as the proxy server or browser. Caching saves time and resources because the server does not have to recreate information, particularly things that demand significant processor time or other resources. The following image contrasts data flow in non-cached and cached environments.



Ektron provides 2 kinds of caching.

- *Caching while logged In* below. Lets you cache part of a Web page; available with some server controls
- *Page-level caching* on the next page. Lets you cache an entire page while not logged in; available with all server controls

Caching while logged In

When a user is logged in, changes appear on the site only after time defined in the cache interval. This reduces the number of database hits, which improves your server's performance. For example, if you add a new item to a Collection, the change

only appears on the site when the cache interval expires. In the meanwhile, use Preview mode to see the updated Collection immediately.

The following server controls can cache individual content while you are logged in.

- Collections
- ContentList
- DhtmlMenu
- ListSummary
- MetadataList
- Menu

IMPORTANT: Caching while logged in does not work with Private content.

To set up caching of content for a Web page:

1. Set the *siteroot*/web.config file's ek_CacheControls property's value to "1".
2. Add to a Web form a server control that supports caching of individual content.
3. Set the control's CacheInterval property to the number of seconds for which data should be cached. For example, to cache for 5 minutes, set CacheInterval to 300.

   The default value is 0 seconds. So, you must change the default to enable caching.

Page-level caching

The following controls can cache individual content while you are not logged in.

- ActiveTopics
- All blog controls
- Collections
- CommunityDocuments
- CommunityGroupBrowser
- CommunityGroupList
- CommunityGroupMembers
- CommunitySearch
- ContentBlock
- ContentList
- DhtmlMenu
- Directory
- Favorites
- FlexMenu
- FormBlock
- Forum
- Friends

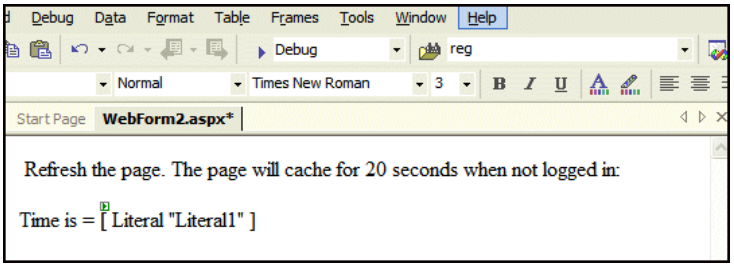
- ListSummary
- MetadataList
- Menu
- Poll
- PostHistory
- SiteMap
- SmartMenu
- TagCloud
- UserProfile

The following example shows a page-level cache for non-logged-in users. You use a server control to define whether a user is logged in, then define if the page is cached, based on the user's status.

1. Create a new Web form in your Ektron project.
2. Add the following text to the Web form:

```
Refresh the page. The page will cache for 20 seconds when not logged in:
Time is =
```

3. Next to **Time is =**, add a literal.



4. Below that, add a ContentBlock server control.
5. Set the DefaultContentID. For this example, `DefaultContentID = 20`.
6. Add the following to the Page_Load event in the code-behind.

```
If Not ContentBlock1.IsLoggedIn Then
  Response.Cache.SetExpires(DateTime.Now.AddSeconds(20))
  Response.Cache.SetCacheability(HttpCacheability.Public)
  Response.Cache.SetValidUntilExpires(True)
  Response.Cache.VaryByParams("id") = True
```

NOTE: On a PageBuilder page, you would use `Response.Cache.VaryByParams ("pageid") = True`.

```
Response.Cache.SetVaryByCustom("cmsCache")
End If
Literal1.Text = Now()
```

7. Build your Web form.
8. Browse to your Web form using a browser.
9. Press your browser's refresh button. If you are not logged in to Ektron, the time remains for twenty seconds. After twenty seconds, when you refresh, the new time appears.

NOTE: You can use the same code in a user control to cache output in a particular region of the page.

Displaying custom XML in Ektron's server controls

Several Ektron server controls have a CustomXml property to add custom XML to a control's generated XML before it is processed by its XSLT. Use the CustomXml property in code-behind with these server controls:

- Cart
- Checkout
- CurrencySelect
- MyAccount
- OrderList
- Product
- ProductList
- ProductSearch
- Recommendation

The following examples shows a C# usage of the CustomXml property in code-behind.

```
protected void Page_Load(object sender, EventArgs e)
       {
       product1.CustomXml =
         "<banner>Save $$$ While Christmas Shopping!</banner>
       <specials><special><link>ProductDemo.aspx?id=1013</link>
       <text>A great gift for Dad!</text></special><special>
       <link>ProductDemo.aspx?id=1015</link><text>
       A great gift for Mom!</text></special></specials>";
       }
```

The following example shows the XML sent to the XSLT file.

```
<root>
  <customXml>
    <banner>Save $$$ While Christmas Shopping!</banner>
       <specials>
          <special>
               <link>ProductDemo.aspx?id=1013</link>
               <text>A great gift for Dad!</text>
          </special>
          <special>
               <link>ProductDemo.aspx?id=1015</link>
               <text>A great gift for Mom!</text>
          </special>
       </specials>
   </customXml>
</root>
```

Learning about Visual Studio

This section provides background information about Microsoft's Visual Studio. For more information, use the help feature installed with Visual Studio and Microsoft's developer Center (Microsoft Visual Studio).

NOTE: The following definitions are from Visual Studio Help.

- **Grid Layout**. Absolute positioning attributes are inserted into elements that are added, and updated in elements that are moved. Elements can be dragged across the Design view surface. The positioning grid and Snap to Grid are available.
- **Flow Layout**. Elements are added without absolute positioning attributes. Web browsers arrange elements in the order that they occur on the page, from top to bottom. You cannot drag elements across the Design view surface or use the positioning grid.

Grid layout is the default, which means that all controls drawn to the Web form in the designer window have absolute positioning. Here is an example.

```
<body MS_POSITIONING="GridLayout">
<form id="Form1" method="post" runat="server">
  <asp:Button id="Button1" style="Z-INDEX: 101;
    LEFT: 160px; POSITION: absolute;
    TOP: 80px" runat="server" Text="Button">
  </asp:Button>
  <asp:Button id="Button2" style="Z-INDEX: 102;
    LEFT: 480px; POSITION: absolute;
    TOP: 88px" runat="server" Text="Button">
  </asp:Button>
  <asp:GridView id="GridView1" style="Z-INDEX: 103;
    LEFT: 208px; POSITION: absolute;
    TOP: 152px" runat="server">
  </asp:GridView>
</form>
</body>
```

In Grid layout, you can position your controls like a WYSIWYG editor with no knowledge of HTML. However, because absolute positioning is not rendered consistently by all browsers, the page layout can be flexible based on the size of other controls on the page, and the Web browser window.

When other controls are dynamically populated, such as a GridView, controls that appear beneath it in the Web form would be obscured if they were positioned absolutely at design time. In addition, when utilizing globalization of pages with different languages, the size of text areas can vary and cause obstructions.

Using Ektron's Developer SDK

The Developer Software Distribution Kit (SDK) for Ektron contains the following components to help you extend and customize your site.

- **Developer API**, which includes:
  - **Server Control API**. An interface for calling the methods and properties of the Ektron server controls. For additional information on the server controls, see Introduction to server controls and the Developer API Documentation.
  - **Web Services API**. Exposes a method's functionality for use with SOAP over HTTP. For additional information on the Web Services, see Web Services and the Developer API Documentation.
  - **.NET Assembly API**. Similar to the Business API provided in previous version, the .NET Assembly API provides an interface for calling the methods and properties that are exposed in Ektron. The Developer API Documentation.
- **Developer API Documentation**

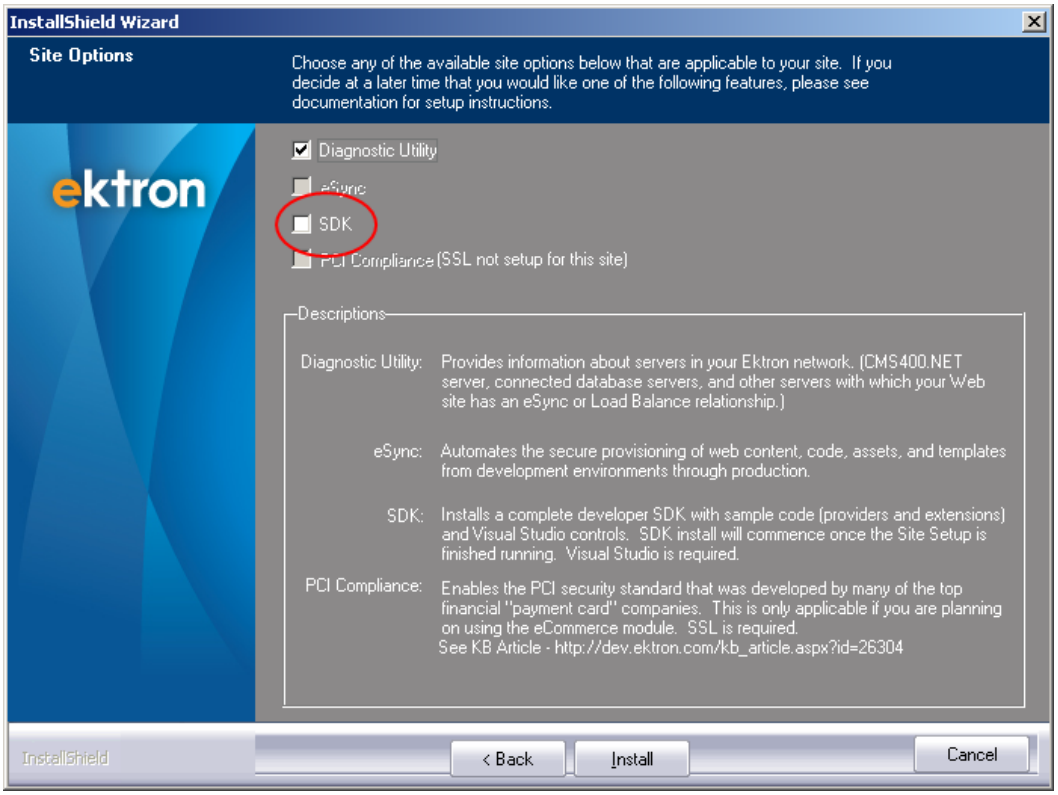
  The API Documentation contains a detailed description of the functions included in each of the APIs. To access the Developer's API documentation in Visual Studio, choose **Help > Contents**. Next, choose Ektron API Documentation from the list of contents. You can also filter the documentation so you see only Ektron's API documentation. Click **Ektron API Documentation** in the filter drop-down box. See also: *Framework API* on page 153.
- **Strategies**

  Strategies are developer-defined software modules that modify the behavior of Ektron.

  NOTE: In versions previous to Ektron version 8.00, developers used the Plug-in Extension Wizard to extend the system. As of version 8.00, Extensions are preferred over the Plug-in system. As of version 9.00, use Strategies. See also:*CMS strategies* on page 2561
- **Ektron server controls Toolbox**

  Ektron controls are installed with the Developer SDK. Server controls let you insert, via drag and drop or programmatically, many standard methods and properties within the Visual Studio environment. This means that you can see the effect of your changes in real time -- you don't have to modify a page then compile a sample project to see the results.

You can install the Developer SDK appears during installation or upgrade of Ektron.



If **Developer SDK** is not checked during installation, you can install it by going to Windows **Start** button **> All Programs > Ektron > CMS400vx*x* > Utilities > CMS400SDK Install**, which runs the `CMS400SDK_setup.exe` installation program.

If you're using Windows 8 or 2012, press the **Windows** key ( )**/Q** then enter **CMS400SDK Install**.

ActiveTopics

8.60 and higher

The ActiveTopics server control displays a forum's most active or most recent topics. The most active are determined by the number of new posts in a topic, or how many replies a post receives.

Inserting the ActiveTopics server control onto a page

PREREQUISITE

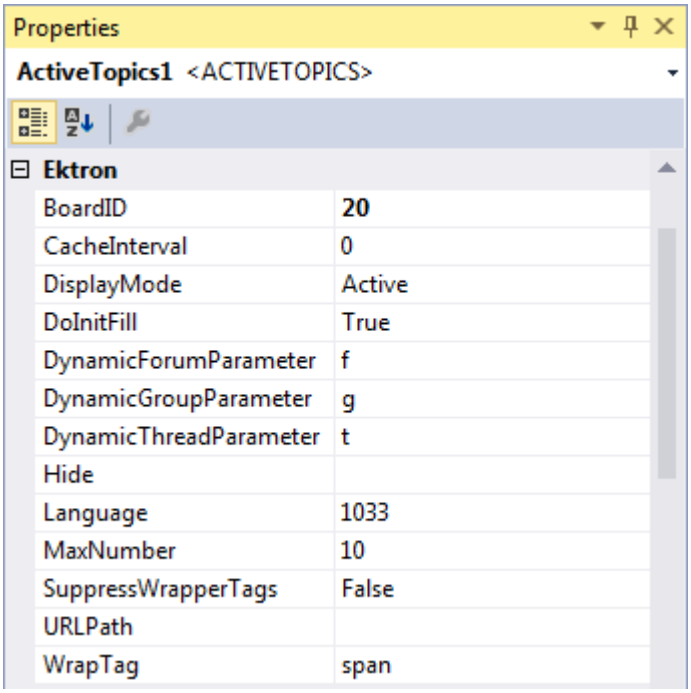
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **ActiveTopics** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:ActiveTopics ID="ActiveTopics1" runat="server" />
```

4. Click on `ActiveTopics` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



ActiveTopic properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **BoardID** (Long)
  The ID of the board from which to get entries. If you don't know the ID, click **Ellipses** (...), then sign in, browse to, and select a discussion board.
- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **DisplayMode** (eDisplayMode)
  Choose between Active or Recent. The default is Active.
  - **Active**. Lists the most active posts.
  - **Recent**. Lists recently added posts.

- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **Hide** (Boolean)
  Select **False** to display this server control on the page. Select **True** to suppress it.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **MaxNumber** (Integer)
  The maximum number of topics listed. The default is **10**.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **URLPath** (String)
  Enter the path to the Forum server control's Web page. For example: `http://<your site>/siteroot/forum.aspx` or `/CMS400Developer/forum.aspx`. If your Forum page and your Active Topics page are in the same folder, just enter the name of the page. For example: `forum.aspx`.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

AnalyticsTracker

8.50 and higher

Track business analytics statistics about visits to your site, such as:

- how often content is viewed
- how many site visitors viewed for first time, and how many returned
- the most popular referral pages

The data recorded by this server control is used by the Most Popular and Trends widgets' **Most Viewed** category. See also: Analyzing Websites

The data recorded is also used by the Classic Analytics Reports, available from the Ektron Workarea **> Reports > CMS Site Analytics** section.

IMPORTANT: Your site license key must support Analytics. You must also set the control's `enableanalytics` property to `true` or set the `enableanalytics` property to `ConfigSpecified`, and the `enableAnalytics` key in your `web.config` file to `true`

Use the ContentIdsList, DefaultContentID, and the DynamicParameter propertiesto determine which content items are viewed when a site visitor browses to a page that contains the control. The properties are additive; that is, they can be used together.

If you want to track your entire site, place an AnalyticsTracker server control on your master pages on your site templates, and use the `DynamicParameter` property.

Inserting the AnalyticsTracker server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

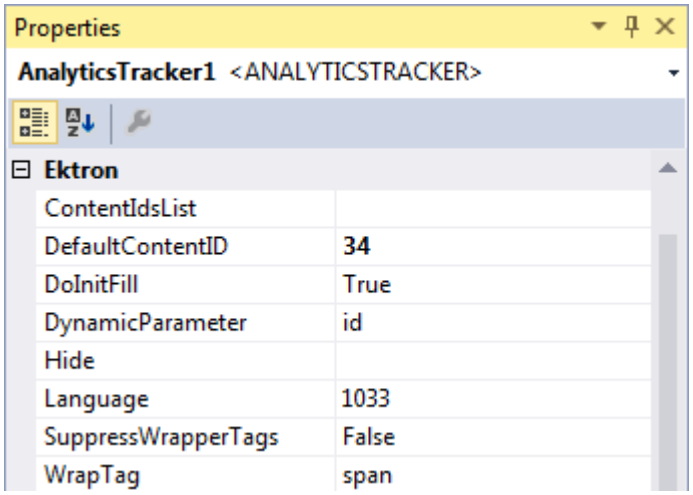
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **AnalyticsTracker** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:AnalyticsTracker ID="AnalyticsTracker1" runat="server" />
   ```

4. Click on `AnalyticsTracker` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as

you modify the property values.



AnalyticsTracker properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **ContentIdsList** (String)

  Enter a comma-separated list of content IDs to be tracked. The DefaultContentId and the dynamicParameter are also tracked. See also: Analyzing Websites

- **DefaultContentID** (Long)

  The ID of a content block being tracked by this server control. It typically would be the content directly above the AnalyticsTracker if no other content block is identified, or is not available. If you don't know the ID number of the content block, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  To make this control dynamic, select **id**. When you do, this server control is attached to the content block passed as a URL parameter.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

AssetControl

8.50 and higher

The AssetControl server control, when viewed on a Web form, displays a drag and drop icon that lets you upload a new asset or update an existing one. When you click this icon, a drag and drop box appears. This box is similar to the upload box in the Workarea. The difference between the Workarea and the server control is, in the Workarea users can only upload assets. With the AssetControl server control, you can upload a new asset or update an existing one by overwriting it. If the asset is overwritten, the previous version is available through Ektron's history feature. See also: Managing Versions of Content.

The appearance of the AssetControl server control can vary depending on your browser. See also: Methods for Importing Assets.

Inserting the AssetControl server control onto a page

PREREQUISITE

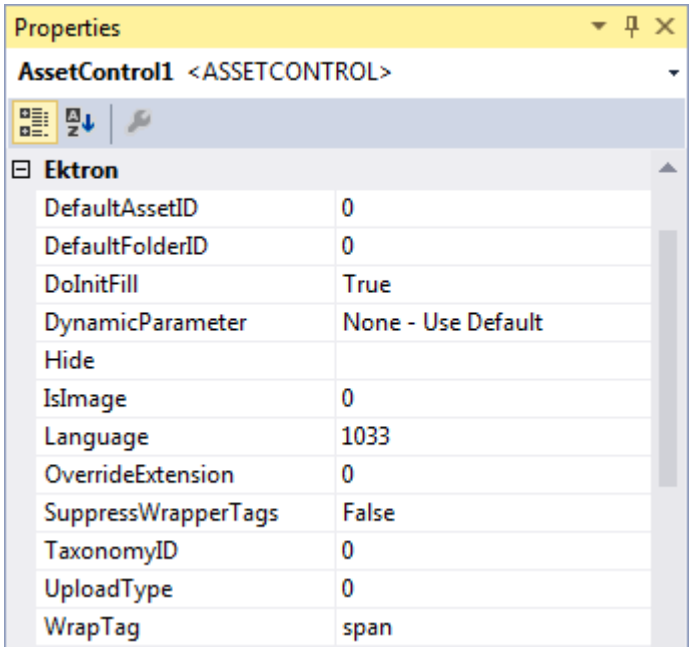
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **AssetControl** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:AssetControl ID="AssetControl1" runat="server" />
```

4. Click on `AssetControl` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



AssetControl properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **DefaultAssetID** (Long)

  The ID of the asset you want to update. This property is used when the `UploadType` property is set to **Update**. If you don't know the ID number of the asset, use the CMS Explorer to browse to it. See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **DefaultFolderID** (Long)

  The ID of the folder where assets are added. This property is used when the `UploadType` property is set to **Add**. If you don't know the ID number of the folder, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a content ID or folder ID dynamically. The content ID is read when the `UploadType` property is set to

Update. The folder ID is read when `UploadType` property is set to Add. To use the default content ID or default folder ID, leave blank.

- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **IsImage** (Integer)
  Setting this control to 1 (one) restricts the control so only images can be uploaded.
  - **1** (one). Restrict the control to uploading images only.
  - **0** (zero). Upload all types of assets.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **OverrideExtension** (String)
  Lets you restrict the type of asset that can be uploaded by its extension. For example, to restrict the control to uploading Word documents, enter **doc** in the property.

  NOTE: When using this property, enter only the extension's letters not the wildcard (*) or the dot (.).

  You can add multiple extensions by creating a comma separated list of extensions. You should limit the list to 5 extensions.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TaxonomyID** (Long)
  The ID of the taxonomy with which to associate the asset.
- **UploadType** (UploadTypeEnum)
  Select whether the control adds new assets or updates existing ones.
  - **Add** (default). Add assets and use the `DefaultFolderID` property. If a file of the same name already exists in the folder, the new file is created using the naming convention `filename(2)`.
  - **Update**. Update assets. In this case, you *must* identify an asset at the `DefaultAssetID` property.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.

- **Div**. Apply attributes to a block of code.
- **Custom**. Lets you use a custom tag.

Blog server controls

8.50 and higher

Ektron provides server controls for displaying a blog on a website. The Blog server control lets you display pieces that typically make up a blog site on a Web form, such as blog entries, posts, subjects, recent posts, RSS feed, archive, calendar, and a blog roll.

Additional Blog server controls let you display each item individually and offer additional ways to customize the layout and appearance. For example, you might display Blog entries and a Blog roll but not a Blog calendar. The individual Blog server controls also let you further define the display details.

NOTE: On a PageBuilder page, you can insert a blog using the Blog widget.

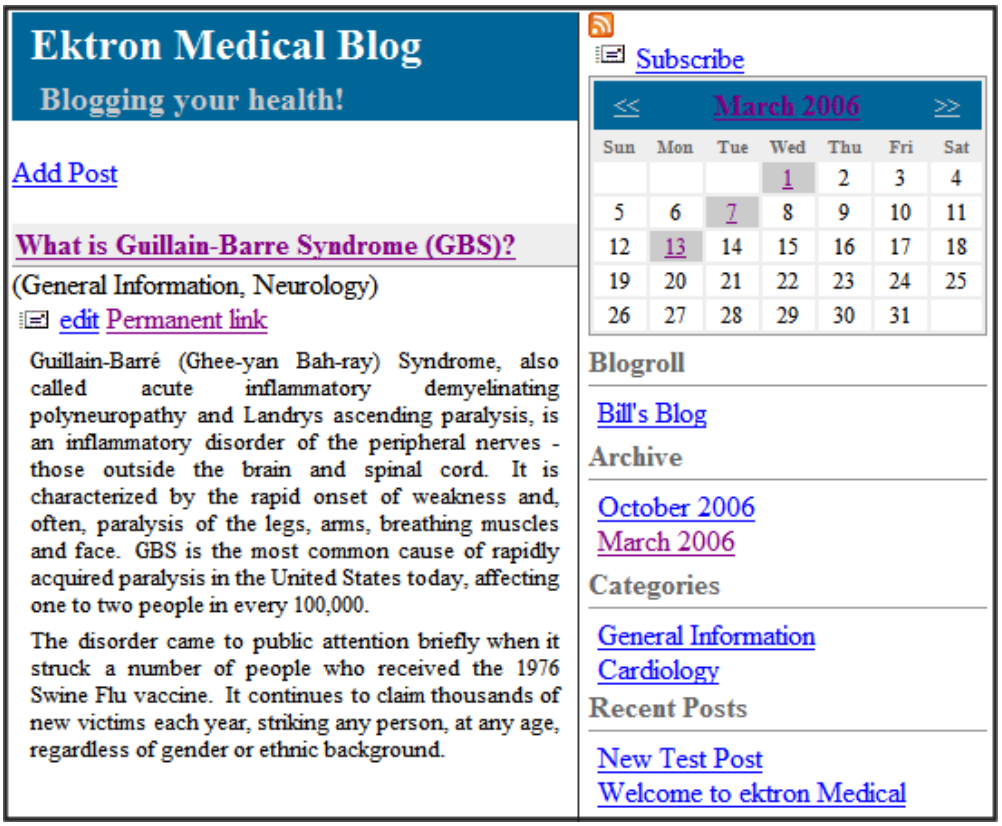
The Blog server controls are as follows:

Blog

8.50 and higher

The Blog server control lets you add a blog to a Web form. This control has the items commonly found on a blog page, such as blog posts, comments link, a blog roll, blog subjects, recent blog posts, the blog's RSS feed, archive, and a blog calendar.

The Blog server control lets you easily maintain the overall look of the blog. Any change to display properties, such as background color or font, affects the entire blog. However, you cannot change the location of each server control item. The title always appears on top, with the tagline below it. The blog posts always appear to the left,and the calendar, blog roll, blog subjects, recent blog posts, RSS feed, archive and blog roll to the right.



NOTE: You can use individual blog server controls to change the page layout.

Inserting the Blog server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

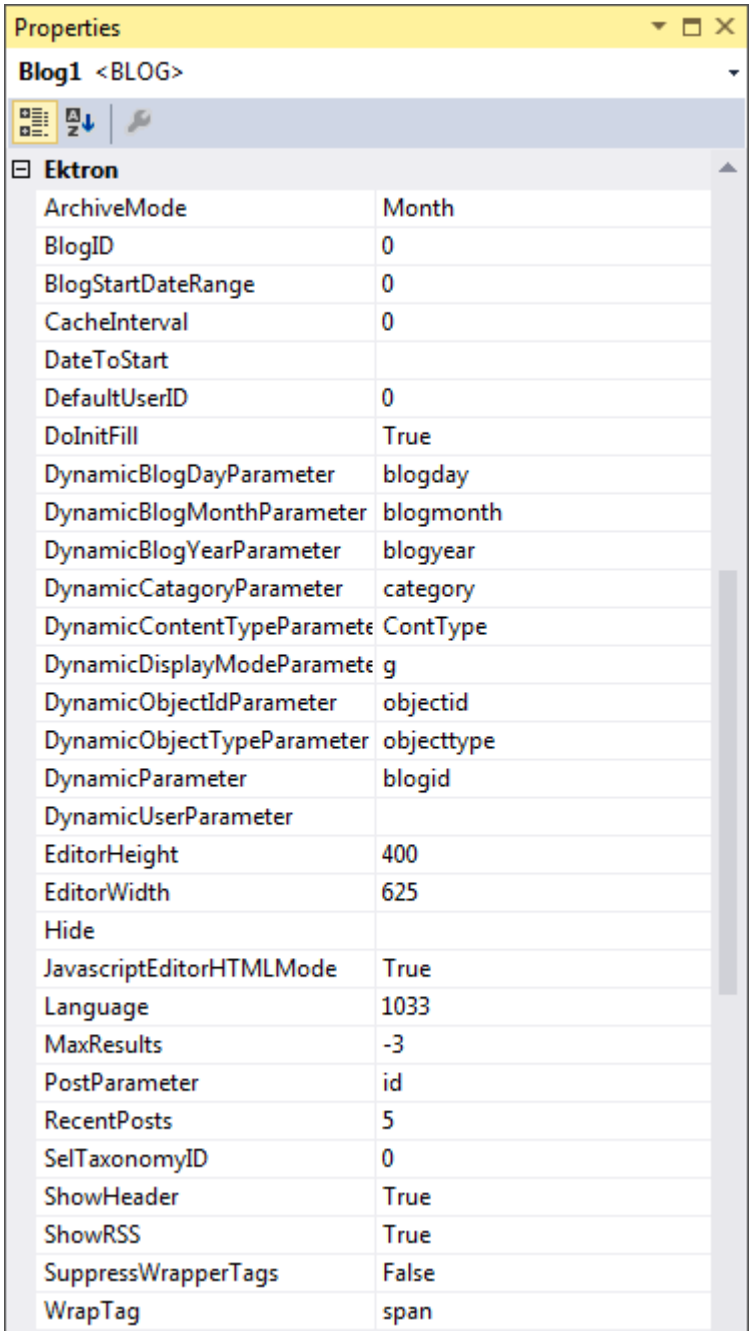
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Blog** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Blog ID="Blog" runat="server" />
```

4. Click on `Blog` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the

property values.

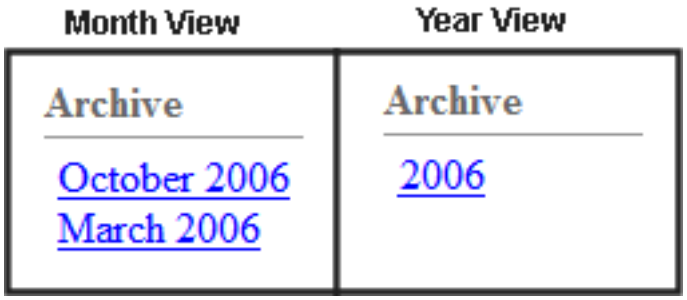


Blog properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **ArchiveMode** (String)

  Select whether the archive appears in *month* format or *year* format. The default is **month**



- **BlogID** (Long)
  The ID of the blog in Ektron. For example: 41. If you don't know the blog ID, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147
- **BlogStartDateRange** (String)
  Set the date range of the Blogs to show. For example, if you want to display blogs for only the past 3 months, set this value to *Quarterly*.
  - **None**. No start date range
  - **Monthly**. Current month
  - **Quarterly**. Past 3 months
  - **BiYearly**. Past 6 months
  - **Yearly**. Past 12 months
- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **DateToStart** (DateTime)
  The date of the most recent blog entries to appear. For example, to display blog entries for January 1, 2012 and before, enter 1/1/2012. Click the drop-down box to access a calendar.
- **DefaultUserID** (Long)
  The ID of the user who owns the blog to be displayed. This property is used when the server control displays a user's blog. To display a blog not associated with a user, leave this property set to 0 (zero) and enter the blog's ID in the `BlogID` property.
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)
  Gets or sets the QueryString parameter to read a Blog ID dynamically. Set to **None. Use Default** if you want to always display the default blog.

- **None. Use Default**. Use the default Blog ID list.
- **ekfrm**. Reads a form block's ID dynamically.
- You may display blogs dynamically by entering any value other than **id**. **id** is the default parameter for **PostParameter**.

- **DynamicUserParameter** (String)
  Gets or sets the QueryString parameter to read a user ID dynamically. Set to **Use Default** if you wish to always display the default user's blog (static).
- **EditorHeight** (Integer)
  Sets the height of the blog editor in pixels.
- **EditorWidth** (Integer)
  Sets the width of the blog editor in pixels.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **MaxResults** (Integer)
  Set the maximum number of posts to display. If set to 0 (zero), there is no limit. If set to

  -1. All posts for the day are shown

  -2. All posts for the month are shown

  -3. Use the **# of Post Visible** setting in the Workarea.

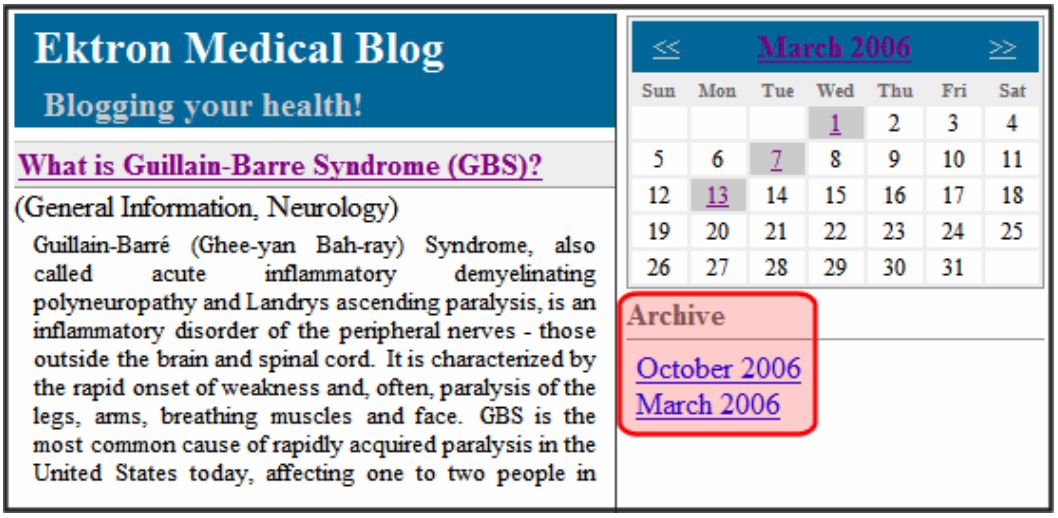
  The default is **-3**.
- **PostParameter** (String)
  Works like the `DynamicParameter` for content blocks. When `id` is selected, this server control passes the blog post ID as a URL parameter. The default setting is **id**.
  - **Blank**. The list of blog posts is static. The links in the blog posts are inactive.
  - **id**. The id of the blog post is passed to the URL as a parameter.
  - **None use default. T**he list of blog posts is static. The links in the blog posts are inactive.
- **RecentPosts** (Integer)
  The number of post links contained in the Recent Posts list. The default is 5.

- **SelTaxonomyID** (Integer)
  Set the ID of the taxonomy with which content is associated if a logged-in site visitor uses the Silver Access Point's **Add HTML Content** option to add content to a Collection server control.
- **ShowHeader** (Boolean)
  - **True** (default). Show title and tagline.
  - **False**. Do not show header and tagline.
- **ShowRSS** (Boolean)
  - **True** (default). Show RSS feed icon ( ).
  - **False**. Do not show RSS feed icon.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

BlogArchive

8.50 and higher

The BlogArchive server control displays past months or years that have blog posts. Use this control with the BlogEntries and Calendar controls to let site users scan for older Blog posts.



When a site visitor clicks a month in the archive, the blog posts for that month appear.

Typically this control appears along side other individual Blog server controls.

Inserting the BlogArchive server control onto a page

PREREQUISITE

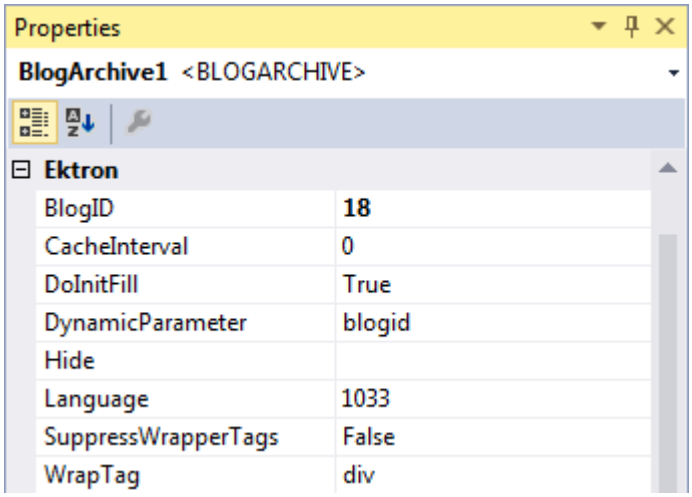
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **BlogArchive** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:BlogArchive ID="BlogArchive1" runat="server" WrapTag="div" BlogID="18" />
   ```

4. Click on `BlogArchive` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



BlogArchive properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **BlogID** (Long)

  The ID of the blog in Ektron. For example: 41. If you don't know the ID of the blog, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)
  Sets the QueryString parameter to read a Blog ID dynamically. Leave blank to always display the default blog.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

BlogCalendar

8.50 and higher

The BlogCalendar server control displays a calendar on a Web page and associates it with a blog. Days with blog posts are highlighted on the calendar. You can use a BlogCalendar server control with a BlogEntries server control to display blog posts for a given day.

Inserting the BlogCalendar server control onto a page

PREREQUISITE

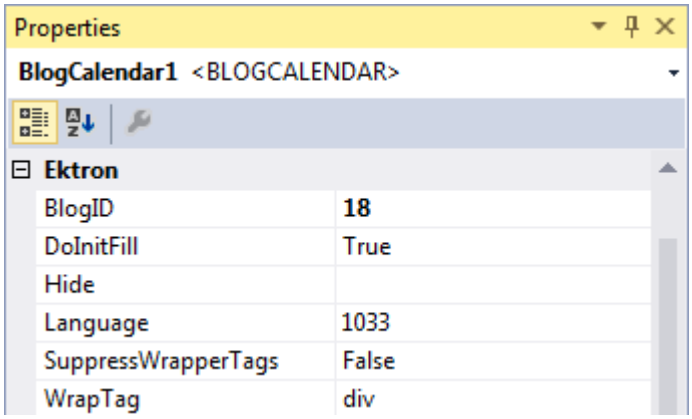
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **BlogCalendar** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:BlogCalendar ID="BlogCalendar1" runat="server" />
```

4. Click on `BlogCalendar` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



BlogCalendar properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **BlogID** (Long)

  The ID of the blog in Ektron. For example: 41. If you don't know the ID of the blog, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

BlogCategories

8.50 and higher

The BlogCategories server control displays a blog's subjects as a clickable list of links on a Web form. When a link is clicked, it displays all subjects associated with the category. Typically, this control appears with other Blog server controls.

Inserting the BlogCategories server control onto a page

PREREQUISITE

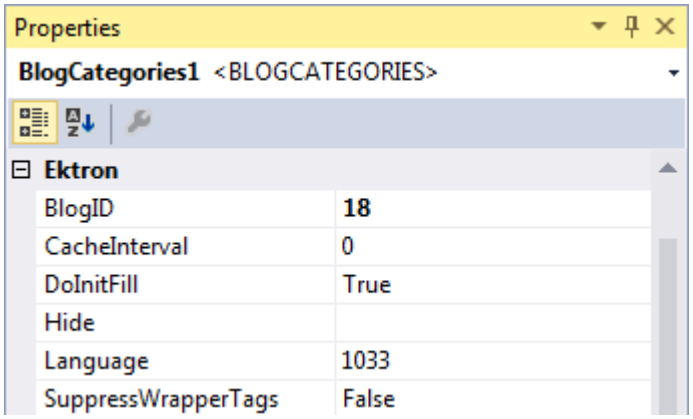
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **BlogCategories** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:BlogCategories ID="BlogCategories1" runat="server" />
   ```
4. Click on `BlogCategories` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as

you modify the property values.



BlogCategories properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **BlogID** (Long)

  The ID of the blog in Ektron. For example: 41. If you don't know the ID of the blog, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **SuppressWrapperTags** (Boolean)

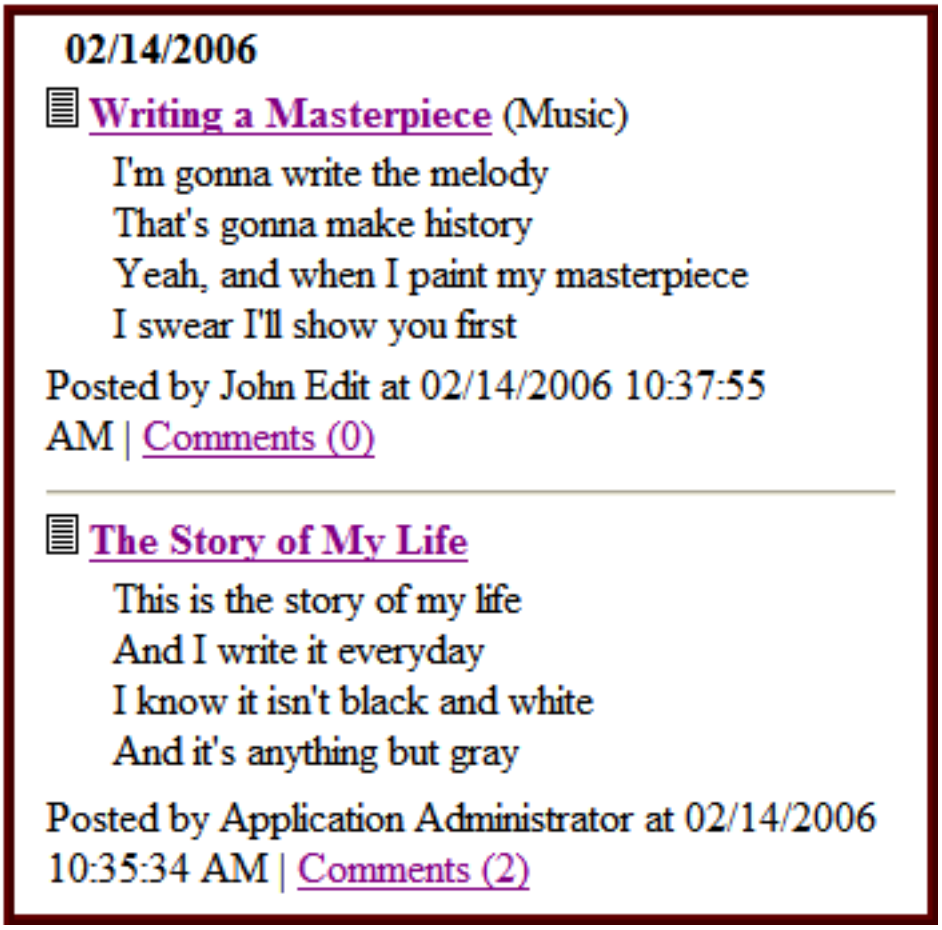
  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

BlogEntries

8.50 and higher

The BlogEntries server control displays several blog posts or a user's Journal (personal blog) on a Web form. You can change this server control's appearance without changing the other blog controls. The following example shows a BlogEntries server control.



To use this server control to dynamically display a *blog's* entries, set the following properties:

- `DynamicParameter`. Set this property to the parameter name used to pass a blog ID to the QueryString. The default is **blogid**.
- `BlogPostParameter`. Set this property to the parameter name used to pass a blog post's ID to the QueryString. The default is **id**.

To use this server control to dynamically display a *user's Journal*, set the following properties:

- `DynamicUserParameter`. Set this property to the parameter name used to pass a user's ID to the QueryString.

- `BlogPostParameter`. Set this property to the parameter name used to pass a blog post's ID to the QueryString. The default is **id**.

Inserting the BlogEntries server control onto a page

PREREQUISITE

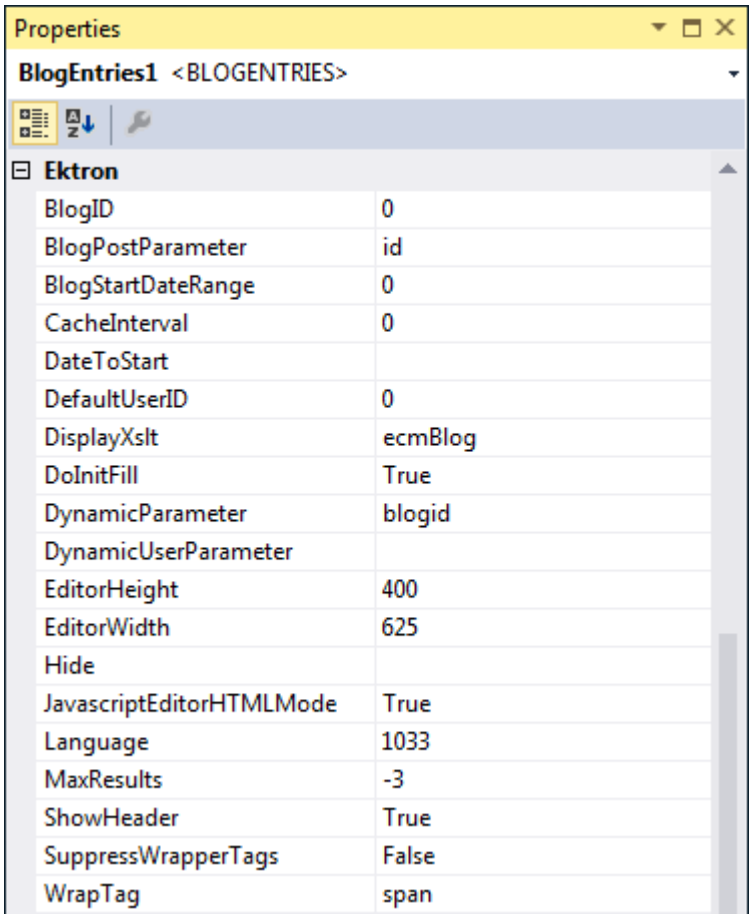
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **BlogEntries** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:BlogEntries ID="BlogEntries1" runat="server" />
```

4. Click on `BlogEntries` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



BlogEntries properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **BlogID** (Long)

  The ID of the blog in Ektron from which blog entries are displayed; for example, 41. This is the default ID that is used when a blog ID is not passed dynamically to the QueryString. To pass a blog ID dynamically, set the `DynamicParameter` property. If you don't know the ID of the blog, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **BlogPostParameter** (String)

  Set this property to the parameter name used to pass a blog post's ID to the QueryString. The default is **id**. When a parameter is defined, this server control passes the blog post's ID as a URL parameter. If you do not set this parameter to **id**, you will not be forwarded to the blog post's page when you click on any links in the post. The default setting is **id**.

  - **Blank**. The list of blog posts is static. The links in the blog posts are inactive.
  - **id**. The ID of the blog post is passed to the URL as a parameter.
  - **None use default. T**he list of blog posts is static. The links in the blog posts are inactive.

- **BlogStartDateRange** (String)

  Set the date range of the Blogs to show. For example, if you want to display blogs for only the past 3 months, set this value to *Quarterly*.

  - **None. N**o start date range
  - **Monthly. C**urrent month
  - **Quarterly. P**ast 3 months
  - **BiYearly**. Past 6 months
  - **Yearly**. Past 12 months

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DateToStart** (DateTime)

  The date of the last blog entries you want to appear. For example, if you want to display blog entries for January 1, 2012 and before, enter 1/1/2012. Click the drop-down box to access a calendar.

- **DefaultUserID** (Long)

  The ID of the user who owns a Journal from which to display journal entries. To display journal entries not associated with a user, leave this property set to 0 (zero) and enter the blog's ID in the `BlogID` property.

  WARNING! If you define a DefaultUserID, it overrides the BlogID property.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a Blog ID dynamically. Set to "None. Use Default" if you want to always display the default blog. The default is **blogid**.

  NOTE: If an ID for this property is passed on the QueryString and a an ID for the DynamicUserParameter property is passed the control displays blog entries for a user.

- **DynamicUserParameter** (String)

  Gets or sets the QueryString parameter to read a User ID dynamically. Set to **Use Default** if you wish to always display the default user's blog (static.)

  NOTE: If an ID for this property is passed on the QueryString and a an ID for the DynamicUserParameter property is passed the control displays blog entries for a user.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **MaxResults** (Integer)

  Set the maximum number of posts to display. The default is **-3**.

  - **0** (zero). No limit
  - **-1**. All posts for the current day
  - **-2**. All posts for the current month
  - **-3**. The **# of Post Visible** Workarea setting

- **ShowHeader** (Boolean)
  Shows the title and tagline when set to True. The default is **True**.
  - **True**. Show title and tagline.
  - **False**. Do not show header and tagline.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

BlogPost

8.50 and higher

The BlogPost server control displays an individual blog post on a page. There are 2 ways that this control displays a blog post.

- If a user is *logged in* as an Ektron or a membership user, the control displays the blog post, comments, and comments from the Web page.
- If a user is *not logged in*, the control displays only the blog post.

NOTE: The **Enable Comments** property in the Workarea must be enabled for comments and the comments form to appear.

Inserting the BlogPost server control onto a page

PREREQUISITE

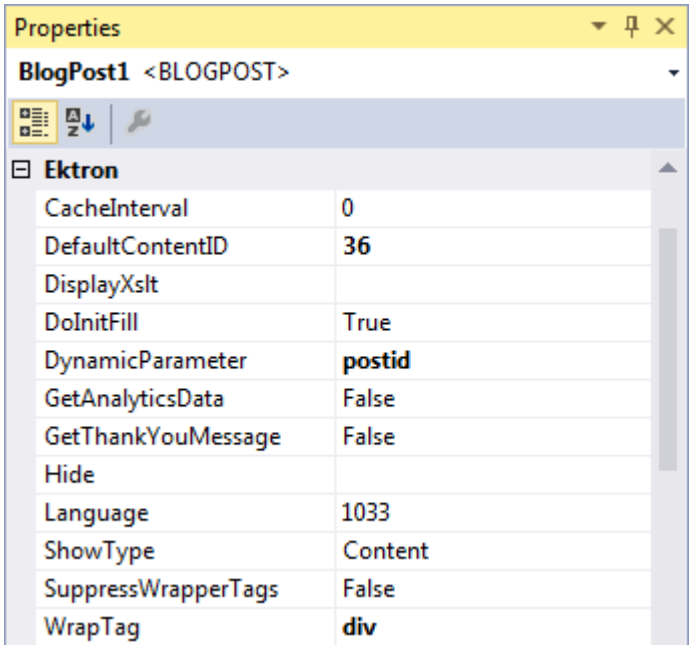
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **BlogPost** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:BlogPost ID="BlogPost1" runat="server" />
```

4. Click on `BlogPost` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



BlogPost properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultContentID** (Long)

  The ID of a default blog post that appears where you inserted this server control if no other content block is identified, or is not available. If you don't know the ID number of the blog post, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)
  To make this blog post dynamic, select **id**. When you do, this server control uses the blog post passed as a URL parameter.
- **GetAnalyticsData** (Boolean)
  Set this property to **True** if you want the following information for each content in the list. Returns **Content View Count**, **Content Rating**, **Content Rating Average**. Create your own XSLT styles to display this data.

  IMPORTANT: This property provides reliable data only when the Business Analytics Feature is on. See Running Ektron Business Analytics.
- **GetThankYouMessage** (Boolean)
  Determines whether a message appears after adding a blog comment.
  - **True**. Displays "Thank you" message after adding a blog comment
  - **False**. Do not display the message.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **ShowType** (String)
  Shows a blog post's content and its comments or only its comments. The default is **Content**.
  - **Content**. Show blog post's content and comments.
  - **Description**. Show comments only.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.

- **Div**. Apply attributes to a block of code.
- **Custom**. Lets you use a custom tag.

BlogRecentPosts

8.50 and higher

The BlogRecentPosts server control displays a list of recent blog post links on a Web form. When you click a link, you are directed to the blog post. The `NumberofPosts` property lets you control how many links appear. Typically, this control appears with other individual Blog server controls.

NOTE: On a PageBuilder page, you can display recent blog posts using the Recent blog posts widget. See also: Creating and Using Widgets

Inserting the BlogRecentPosts server control onto a page

PREREQUISITE

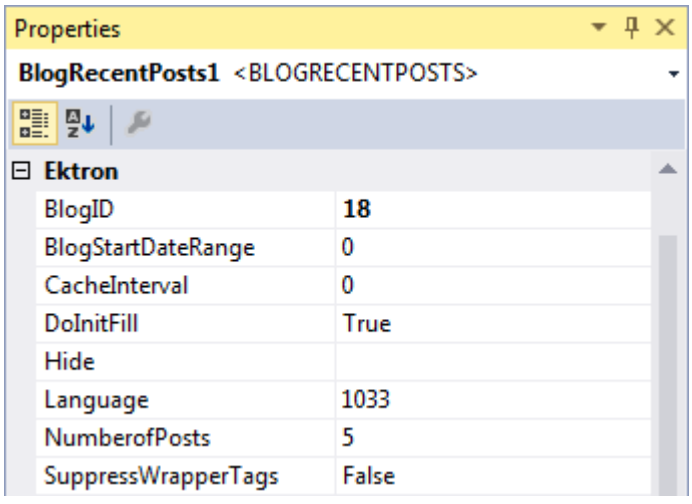
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **BlogRecentPosts** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:BlogRecentPosts ID="BlogRecentPosts1" runat="server" />
```

4. Click on `BlogRecentPosts` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



BlogRecentPosts properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **BlogID** (Long)
  The ID of the blog in Ektron. For example: 41. If you don't know the ID of the blog, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147
- **BlogStartDateRange** (String)
  Set the date range of the Blogs to show. For example, want to display blogs for the past 3 months, set to*Quarterly*.
  - **None**. No start date range
  - **Monthly**. Current month
  - **Quarterly. P**ast 3 months
  - **BiYearly**. Past 6 months
  - **Yearly**. Past 12 months
- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **NumberofPosts** (Integer)
  Sets the number of post links to display. The default is 5.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

BlogRoll

8.50 and higher

The BlogRoll server control displays a *blog roll* on a Web form. A blog roll is a list of links to other blogs. Typically, this control appears with other individual Blog server controls.

Inserting the BlogRoll server control onto a page

PREREQUISITE

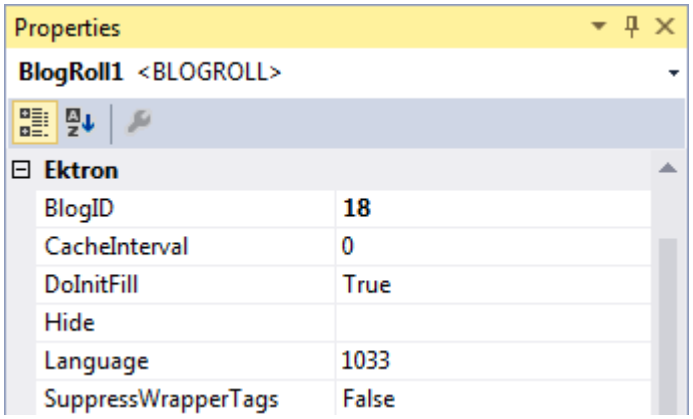
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **BlogRoll** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:BlogRoll ID="BlogRoll1" runat="server" />
```

4. Click on `BlogRoll` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



BlogRoll properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **BlogID** (Long)

The ID of the blog in Ektron. For example: 41. If you don't know the ID of the blog, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

BlogRSS

8.50 and higher

The BlogRSS server control displays a blog's RSS feed icon ( ) on a Web form. When the icon is clicked, the blog's RSS feed appears. Typically this control appears along side other individual Blog server controls.

Inserting the BlogRSS server control onto a page

PREREQUISITE

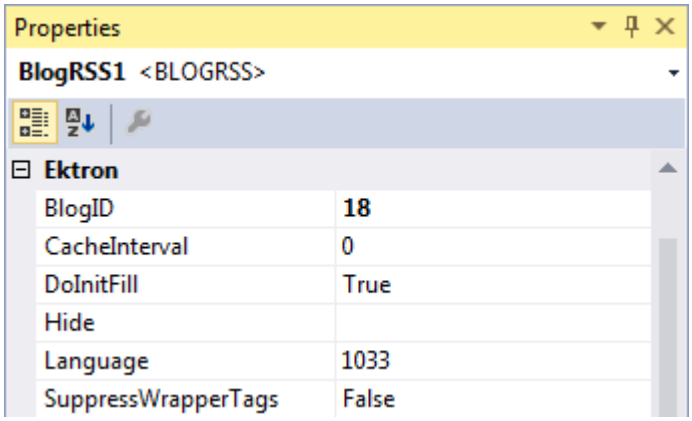
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **BlogRSS** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:BlogRSS ID="BlogRSS1" runat="server" />
```

4. Click on `BlogRSS` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



BlogRSS properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **BlogID** (Long)

  The ID of the blog in Ektron. For example: 41. If you don't know the ID of the blog, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

BreadCrumb

8.50 and higher

Your website is made up of Web forms. Each page depends on a form to determine much of its appearance. Forms and pages have a parent > child relationship. That is, a form can be associated with any number of pages. When you define breadcrumb properties, you define them for a *form*. All pages that use a form inherit its breadcrumb properties.

Assume, for example, that a Web form is used for the Human Resources section of your website. You might use the title **Human Resources** to identify the form in the BreadCrumb server control. Whenever a page is visited that uses that form, **Human Resources** appears on the breadcrumb trail -- that is, the form title appears, not the individual page.

Inserting the Breadcrumb server control onto a page

PREREQUISITE

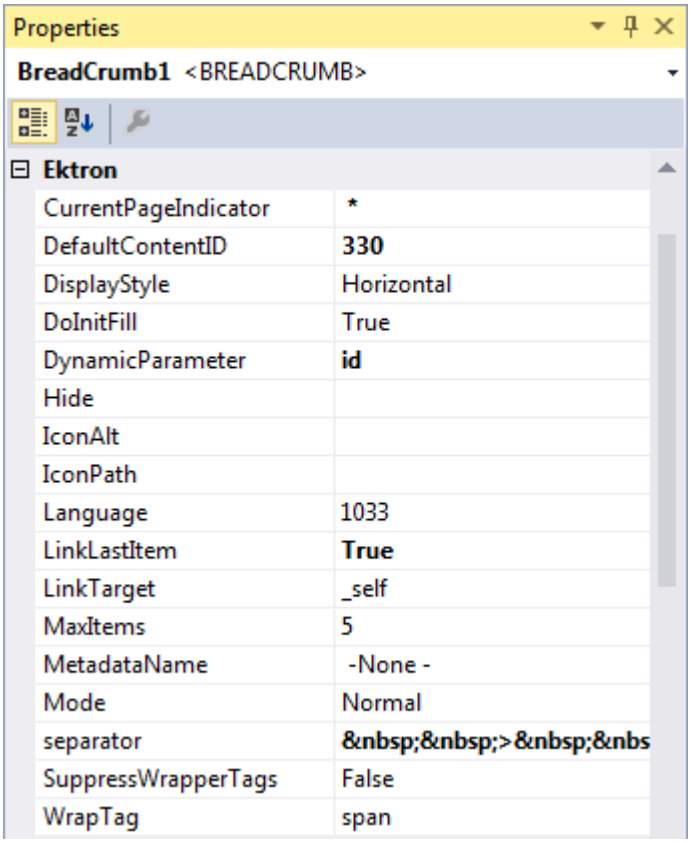
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Breadcrumb** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:Breadcrumb ID="Breadcrumb1" runat="server" />
   ```
4. Click on `Breadcrumb` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



BreadCrumb properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CurrentPageIndicator** (String)

  If desired, enter one or more symbols or characters that identify the current page in the breadcrumb trail. See example below.

  These characters appear *after* the image or title that identifies the current page in the breadcrumb trail.

- **DefaultContentID** (Long)

  Gets or Sets the ContentID for the BreadCrumb display title.

- **DisplayStyle** (String)

  Indicate how to display the breadcrumb trail: horizontally or vertically. The default is Horizontal.

- **DisplayTitle** (String)

  IMPORTANT: This property is for code-behind only and does not show up in the Visual Studio Properties window. It cannot be set in design time.

  Enter text to describe this Web form in the breadcrumb trail. For example, if the Web form's properties you are defining is used for all Human Resources pages on your site, enter **Human Resources**.

  If you define an image in the `IconPath` property below, the image appears in the trail, followed by this text.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a content ID dynamically.

- **Hide** (Boolean)

  Used to hide the breadcrumb trail in design time and run time.

  - **True**. Hide breadcrumb trail
  - **False**. Show breadcrumb trail

- **IconAlt** (String)

  If you define an image in the `IconPath` property, enter "Alt" text to appear if a site visitor hovers the cursor over that image. Here is an example.

- **IconPath** (String)

  If you want the breadcrumb trail to display an image to identify this Web form, enter the path to the image. For example: *siteroot*`\Workarea\Images\bc_meta_icon.gif`

  IMPORTANT: The image location must be relative to the Web root.

On the breadcrumb trail, the image precedes any text defined in the `DisplayTitle` property.

WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LinkLastItem** (Boolean)

  Use this property to determine whether the last breadcrumb item appears as a hyperlink on this Web form. If this property is set to **True**, and a user clicks the item, the current page reappears.

  - **True**. Last item is a hyperlink
  - **False**. Last item is an image or text only; the user cannot click on it

- **LinkTarget** (String)

  Determines the type of window that appears when you click a link in the server control.

  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.

- **MaxItems** (Integer)

  Enter the maximum number of items in the breadcrumb trail on this Web form. The default is **5**. If you set a value of 1 or greater and the user navigates beyond that number of pages, only the most recent pages appear. The older pages disappear from the trail. 0 (zero) = unlimited.

- **MetadataName** (String)

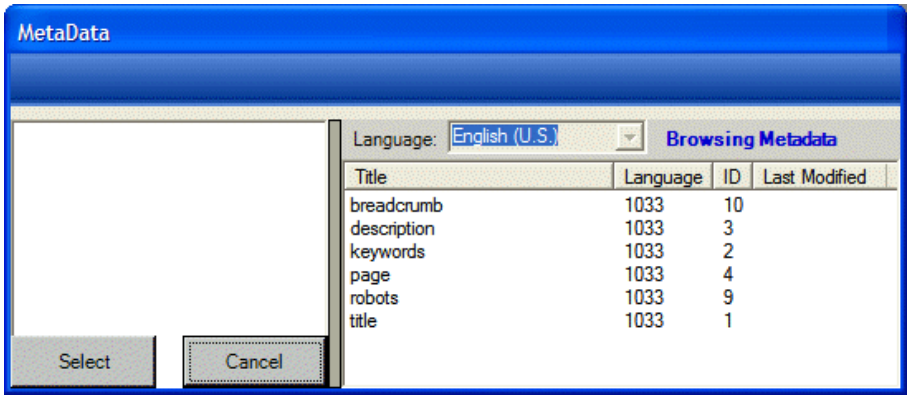
  Specify the name of a Metadata Type that you want to associate with the page.

BreadCrumb metadata type

To associate a metadata type with the BreadCrumb server control:

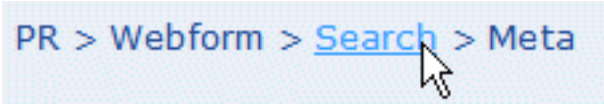
1. In the properties window for the BreadCrumb server control, click on the **MetadataName** property.
2. Click **Ellipses** ().
3. If you are not logged in, log in using the CMS Explorer window.

4. The Browsing Metadata screen appears.

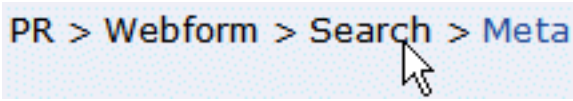


5. Select a Metadata Type to apply to the BreadCrumb server control.
6. Set the DynamicParameter to ID. This allows the metadata to reflect the content block on the page.

- **Mode** (String)
  Lets you display the breadcrumb trail as non-hyperlinked, plain text. The default is **Normal**.
  - **Normal**. Breadcrumb trail is hyperlinked

  - **DisplayOnly**. Breadcrumb trail is plain text

- **Separator** (String)
  Enter one or more characters to separate breadcrumb trail items on this Web form. The default character is the greater than sign (**>**).

- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
- Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Determining a breadcrumb trail's appearance

For each Web form, you can customize the breadcrumb trail. For example, you might want the breadcrumb trail to appear horizontally on one Web form and vertically on another. The following property list determines the breadcrumb trail's appearance.

- **CurrentPageIndicator**. Symbols or characters to identify the current page.
- **DisplayStyle**. Whether it appears horizontally or vertically.
- **LinkLastItem**. Whether the last item is a hyperlink.
- **LinkTarget**. The type of window that appears when a user clicks an item.
- **MaxItems**. The maximum number of items.
- **Mode**. Whether the breadcrumb trail appears as linked text or plain text.
- **Separator**. Symbols or characters that separate items.

Determining how form pages appear on breadcrumb trail

Use these properties define how any page that uses this form appears within a breadcrumb trail. It does not matter where the trail appears.

Note that you can use text, an image, or both to describe the form within the breadcrumb trail. If you use both, the image appears first, followed by the text. The following property list determines how the Web form appears on a breadcrumb trail.

- **DisplayTitle**. Text to describe it.
- **IconAlt**. "Alt" text associated with image specified in IconPath property.
- **IconPath**. Image to define it.

Creating a horizontal or vertical breadcrumb trail

You can display a breadcrumb trail vertically or horizontally on a Web page with the **DisplayStyle** parameter. In addition, you can define how many breadcrumbs (**MaxItems**) are left for site visitors to navigate back. You can also use an image and Alt text instead of, or in addition to, text to display the breadcrumb trail. The foillowing examples show the variations and are described in *BreadCrumb properties* on page 2197.

NOTE: If a site visitor revisits a page that is in the breadcrumb trail already, the breadcrumb trail reverts back to that point in the trail. For example, if you visit the following pages **Dev > FAQ > WebForm > PR**, and then return to FAQ, the breadcrumb trail looks like this: **Dev > FAQ**.

- Horizontal; text only

  Dev > WebForm > pr > FAQ > Meta*

- Horizontal with images

- Horizontal with images and Alt text

- Vertical; text only

- Vertical with images

- Vertical with images and Alt text

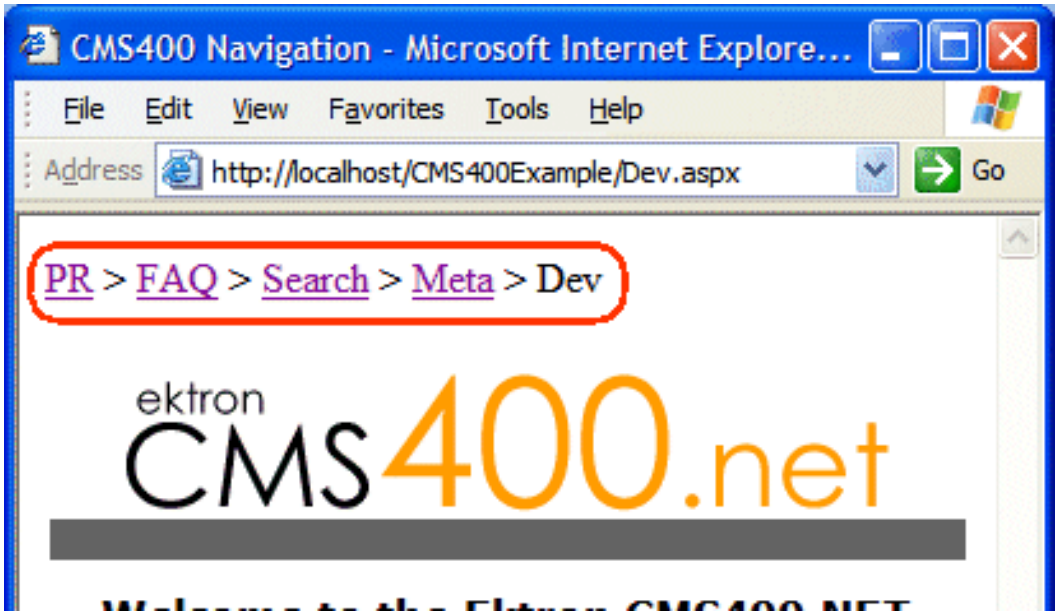
To implement sitemap breadcrumb navigation, add a BreadCrumb server control to every Web form in your site. Doing this ensures that a breadcrumb trail may appear on each Web page.

Using the BreadCrumb server control

NOTE: Do not add a BreadCrumb server control to a Web form that contains a Calendar server control. When both controls appear on a page and a site visitor clicks on different months, the BreadCrumb trail could look like this: Calendar > Calendar > Calendar > Calendar > Calendar. This happens because each time a site visitor clicks on a month, you are in effect opening a new Web form.

Add a BreadCrumb server control to each Web form for which you want to create a breadcrumb trail. To use the BreadCrumb server control:

1. Open a Web form for which you want to create a breadcrumb trail.
2. Drag and drop the BreadCrumb server control onto the Web form.
3. Set the `DisplayTitle` property.
4. Save the Web form.
5. Repeat steps 1 through 4 for all Web forms in the breadcrumb trail.
6. Open a browser.
7. View a Web page with a BreadCrumb server control in it.
8. Navigate to the rest of the Web pages that contain BreadCrumb server controls.
9. As you move around, the pages are added to the breadcrumb trail.



Displaying a content block's title in the breadcrumb trail

By adding a line or 2 of code, the breadcrumb trail can display a content block's title. You can add a line of code to each Web form. Or, if your content is dynamic, add the code once and, each time a new piece of content is called dynamically, a new breadcrumb is created.

You can only use one content block per Web form with the breadcrumb. If a page has multiple content blocks, select one that best describes the page.

1. Drag and drop a **ContentBlock server control** on a Web form.
2. Choose a **DefaultContentID** for the content block.
3. Drag and drop a **BreadCrumb server control** on the same form.
4. Remove the word **Title** from the `DisplayTitle` property.
5. Add the following line of code to the pageload event code-behind:

```
BreadCrumb1.DisplayTitle = ContentBlock1.EkItem.Title
```

6. Build the project.
7. View the Web form in a Web browser.

The following example content block shows titles used as breadcrumbs.

The word Title appears in the breadcrumb trail when the `DisplayTitle` property is left empty. Even with the added code, if the breadcrumb loads before the content block, the breadcrumb has no information in the `DisplayTitle` property and therefore displays the word title by default.

If the word Title appears in the breadcrumb trail, check to make sure the word Title does not appear in the `DisplayTitle` property. Next, if Title still appears, add the following line of code to the page load event of the code-behind:

```
ContentBlock1.Fill()
```

The code in the page load event should now look like this:

```
ContentBlock1.Fill()
BreadCrumb1.DisplayTitle = ContentBlock1.EkItem.Title
```

This ensures that the content block information is loaded first.

BusinessRules

8.50 and higher

The BusinessRules server control lets you add a Business Rule created in the Workarea to a Web form. It also lets you add a place holder for a Business Rule. Then, an Ektron administrator can create a business rule at a later date.

Inserting the BusinessRules server control onto a page

PREREQUISITE

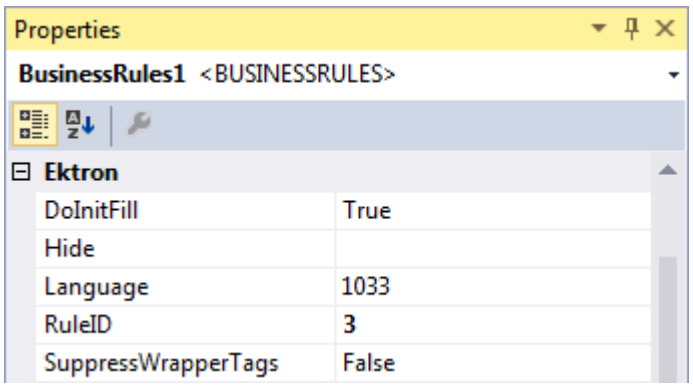
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **BusinessRules** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:BusinessRules ID="BusinessRules1" runat="server" />
```

4. Click on `BusinessRules` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



BusinessRules properties

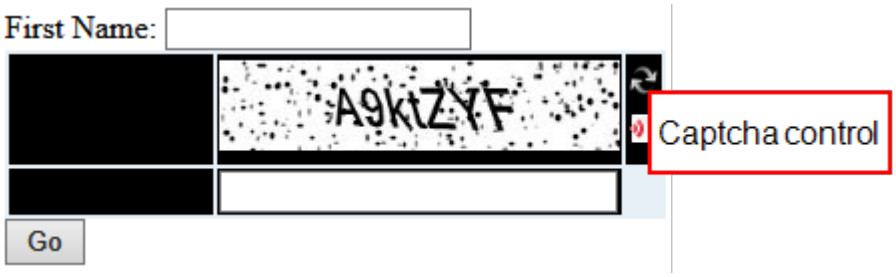
The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **RuleID** (Long)
  The ID of the Ruleset to evaluate. If you don't know the ID number of the Ruleset, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Captcha

9.00 and higher

The Captcha server control lets an Ektron developer place a Captcha control on any web form. The Captcha control is launched when a site visitor clicks the form's submit button. To submit the form, the site visitor must enter characters that match the displayed characters.



The following example shows a `<body>` tag with a Captcha control. In addition to placing the Captcha server control, you must create the form and submit button.

```
<body>
   <form id="form1" runat="server">
      First Name: <asp:TextBox ID="fname" runat="server"></asp:TextBox>
    <div>
    <cms:Captcha ID="captcha1" runat="server" />
    </div>
    <asp:Button Text="Go" ID="button1" runat="server" OnClick="button1_Click" />
    </form>
</body>
```

Inserting the Captcha server control onto a page

PREREQUISITE

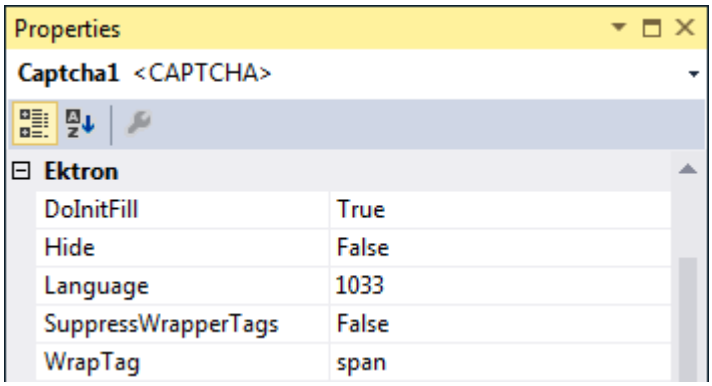
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Captcha** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Captcha ID="Captcha1" runat="server" />
```

4. Click on `Captcha` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



Styling the Captcha Control

To style the Captcha control, enter a pair of `<style>` tags into the page's header, as shown below.

```
<head runat="server">
     <title></title>
       <style type="text/css" >
           #captcha1 table td { background-color:black;}
        </style>
</head>
```

Captcha properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Collection

8.50 and higher

IMPORTANT: Starting from release 8.60, the Collection server control was replaced by the FrameworkUI: <ektron:ContentView> templated server control. If you are already using the Collection server control, you can continue to do so, but Ektron recommends using current versions of functionality.

The Collection server control displays a collection that you create in the Workarea. The control lets you customize the display of the collection on a Web page.

When added to a template and visited, collections can look like the following illustration, which shows 3 collections on the same page of a sample site. You can modify the display by editing its properties.

NOTE: On a PageBuilder page, you can insert a collection using the Collection widget. See also: Creating and Using Widgets

Inserting the Collection server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

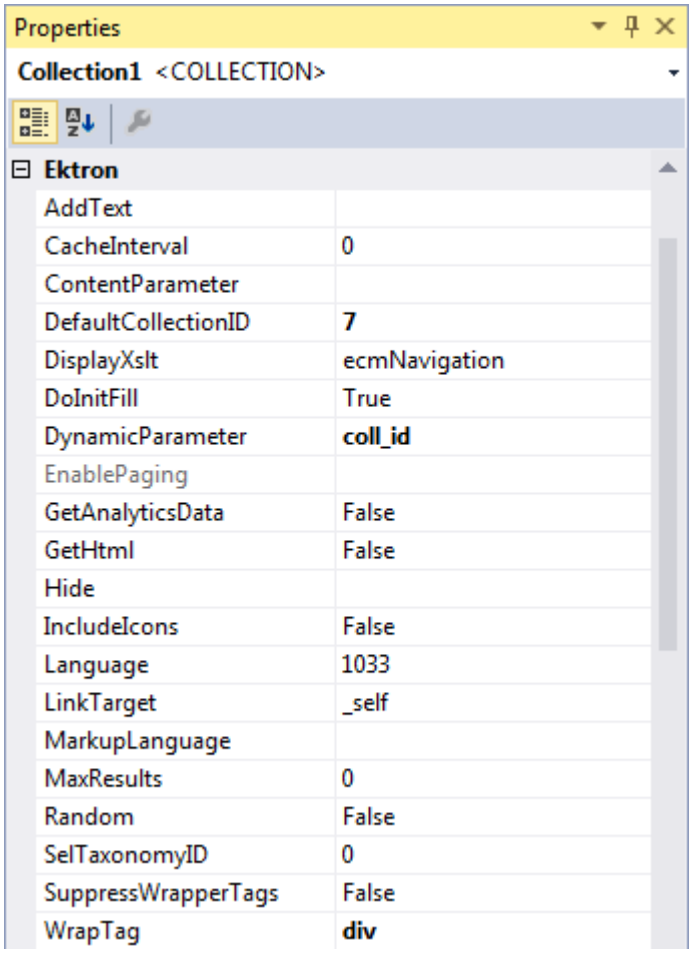
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.

3. Drag the **Collection** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Collection ID="Collection1" runat="server" />
```

4. Click on `Collection` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.

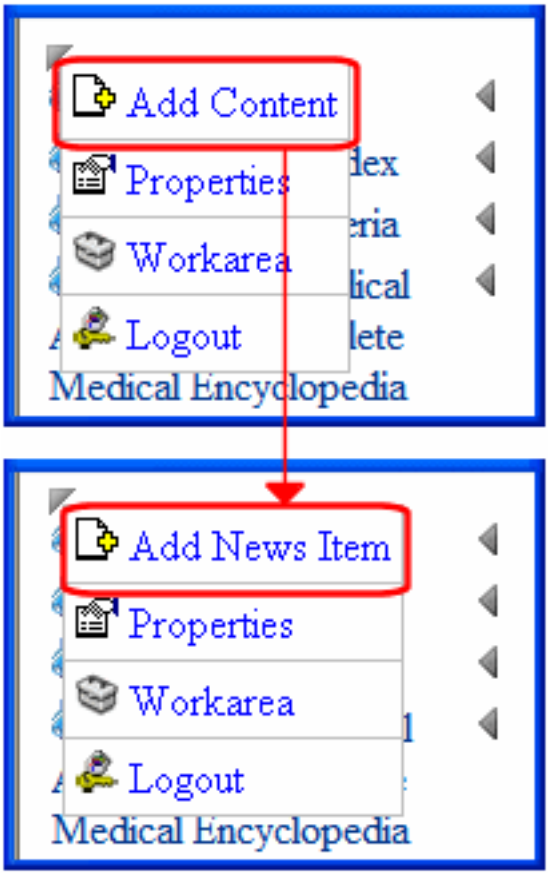


Collection properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AddText** (String)

  Override the control's default text for the Add Content menu item. For example, If you have a News website, you could change **Add Content** to **Add News Item**.



- **CacheInterval** (Double)
- The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

  NOTE: If the `EnablePaging` property is set to True, the `CacheInterval` property is disabled.

- **ContentParameter** (String)

  Checks the QueryString for this value and replaces the collection with a content block when specified. Leave blank to always display the Collection.

- **DefaultCollectionID** (Long)

  The ID of a collection that appears where you insert this server control if no other collection is identified, or is not available. If you don't know the ID number of the collection, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **DisplayXslt** (String)

  Determines how the information on the page appears.

  - **None**. Databind only
  - **ecmNavigation**. Lists the title of every content block in the collection

- **ecmTeaser**. Lists the title of every content block in the collection plus the content summary

  See also: *ecmTeaser Display example* on page 2215.

- **Path to Custom Xslt**. If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

WARNING! If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.

- **DoInitFill** (Boolean)

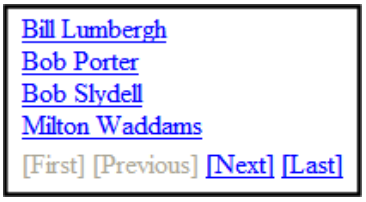
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  To make this collection dynamic, select **coll_id**. When you do, this server control uses the collection passed as a URL parameter.

- **EnablePaging** (Boolean)

  This property, in conjunction with the **MaxNumber** property, lets site visitors view an unlimited number of items while controlling the amount of screen space. The **MaxNumber** property limits the number of items displayed. If you set this property to **True**, and the number of items exceeds **MaxNumber**, navigation aids appear below the last item, allowing the visitor to go to the next screen.See the following example.

  For example, if a collection has 9 items and the `MaxResults` property is set to 4, the screen displays only the first 4 items. When the site visitor clicks **[Next]**, he sees items 5, 6, 7 and 8, and so on.

NOTE: If the `EnablePaging` property is set to True, the `CacheInterval` property is disabled.

- **GetAnalyticsData** (Boolean)

Set this property to **True** if you want the following information for each content in the list. Returns **Content View Count**, **Content Rating**, **Content Rating Average**. Create your own XSLT styles to display this data.

IMPORTANT: This property provides reliable data only when the Business Analytics Feature is on. See Running Ektron Business Analytics.

- **GetHtml** (Boolean)

  Set to **True** if you want to retrieve and display content (html body) for all content blocks in the collection. For example, to display content inside a Web server control such as a GridView.

- **Hide** (Boolean)

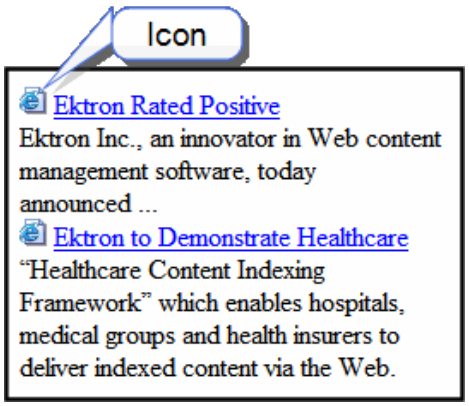
  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **IncludeIcons** (Boolean)

  Choose whether to display icons next to the collection list's links.

IMPORTANT: This property only works if ecmSummary or ecmTeaser is used in the `DisplayXslt` property. If the `[$ImageIcon]` variable is used in an EkML file and that file is assigned to the `MarkupLanguage` property, this property acts as True.



- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LinkTarget** (ItemLinkTargets)

  Determines the type of window that appears when you click a link in the server control.

  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.

- **MarkupLanguage** (String)

  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.

  The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.

  To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.

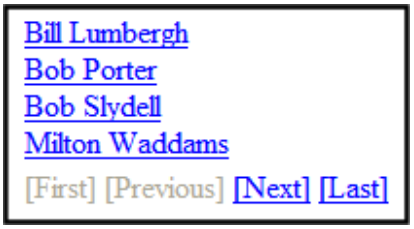
  See also: *Ektron Markup Language* on page 2633

  See also: *collection.ekml* on page 2634

  NOTE: If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored. If the EkML file contains the `[$ImageIcon]` variable, the `IncludeIcons` property acts as True.

- **MaxResults** (Integer)

  Enter the maximum number of items to appear in the initial display of this server control. To set no maximum, enter zero (0). To let site visitors view more than the maximum but limit the amount of space, enter the maximum number of results per page here. Then, set the **EnablePaging** property to **True**. If you do and more than the number of **MaxResults** are available, navigation aids help the site visitor view additional items. See example below.

- **Random** (Boolean)

  Set to **True** if you want to randomly display one collection item. The item changes each time a site visitor views the page.

  NOTE: If you use a custom XSLT or EkML file, the type of content displayed can be manipulated. For example, if you use an EkML file that has the `[$Html]` variable in it, the actual content appears instead of a link. See also: *Ektron Markup Language* on page 2633 and *[$Html]* on page 2666.

- **SelTaxonomyID** (Integer)

  Set the ID of the taxonomy with which content is associated if a logged-in site visitor uses the Silver Access Point's **Add HTML Content** option to add content to a Collection server control.

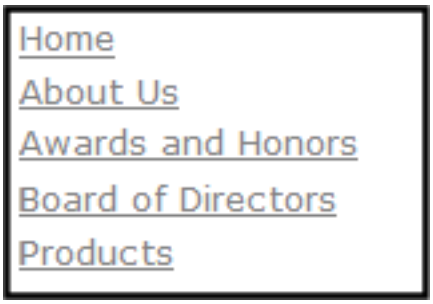
- **SuppressWrapperTags** (Boolean)

  This property is set to `false` because Ajax uses `<div>` tags to rewrite the region around the tag. You *cannot* change the value to `true`.

- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

ecmNavigation Display Example

The following figure shows a collection being used as a navigation menu.



ecmNavigation XSL code

The following XSL code creates the ecmNavigation Display. You can use this code as the basis to design your own XSLT.

WARNING! If you create a custom file, do not store it in the Workarea folder. If you do, the file will be overwritten when you upgrade.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <table border="0" cellspacing="0" cellpadding="0" width="100%">
   <xsl:for-each select="Collection/Content">
   <tr>
      <td>
         <a>
            <xsl:attribute name="href">
                <xsl:value-of select="QuickLink"/>
            </xsl:attribute>
                <xsl:value-of select="Title"/>
         </a>
     </td>
   </tr>
   </xsl:for-each>
  </table>
</xsl:template>
</xsl:stylesheet>
```

ecmTeaser Display example

The following example shows a collection using the ecmTeaser display style.

ecmTeaser XSL code

The following XSL code creates the ecmTeaser Display. You can use this code as the basis to design your own XSLT.

WARNING! If you create a custom file, it is strongly recommended to not store the file in the Workarea folder. If you do, the file will be overwritten when you upgrade.

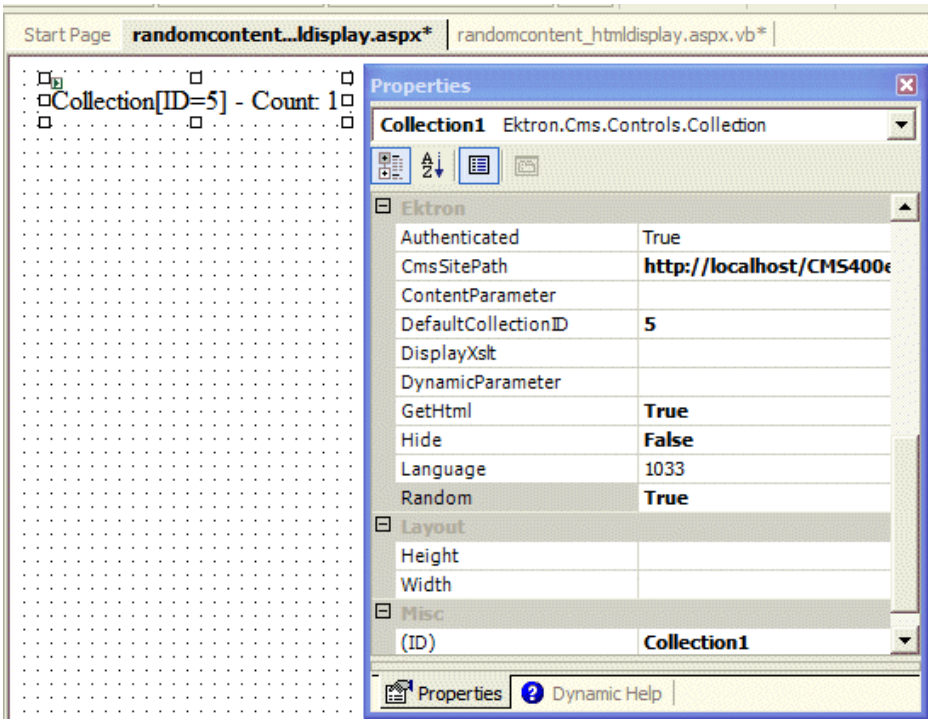
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <table border="0" cellspacing="0" cellpadding="0" width="100%">
    <xsl:for-each select="Collection/Content">
      <tr>
        <td>
          <a>
            <xsl:attribute name="href">
              <xsl:value-of select="QuickLink"/>
              </xsl:attribute>
             <xsl:value-of select="Title"/>
          </a> 
        </td>
      </tr>
      <tr>
         <td>
            <xsl:value-of select="Teaser" />
         </td>
      </tr>
      <tr>
          <td> </td>
      </tr>
    </xsl:for-each>
  </table>
</xsl:template></xsl:stylesheet>
```

Using the Collection server control Programmatically example

The following is an example of using code-behind, and a drag and drop Collection server control to display random content from a collection.

1. Drag and drop a Collection server control on your Web form.
2. Set the properties in the properties window.

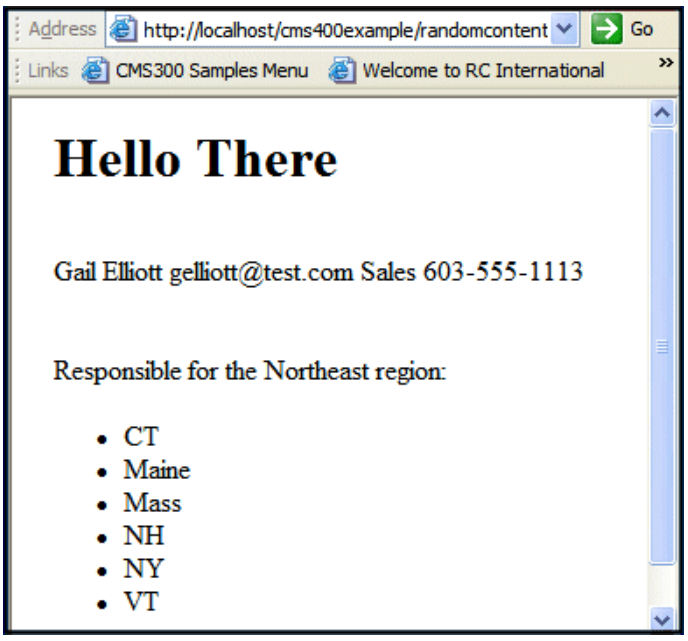
NOTE: In this example, the `Random` and `GetHtml` properties must be set to **True**.



3. Add the following code to the code-behind.

```
Dim str As String
str = "<h1>Hello There</h1><br>"
str &= Collection1.EkItems(0).Html
Collection1.Text = str
```

4. Build and browse your Web form.



Retrieving the XML Structure of a Collection

Retrieving the XML structure of a collection allows for greater control over developing XSLs. The following is an example of how to do that.

1. Open a new Web form.
2. Drag and drop a Collection server control onto it.
3. Set the `DefaultCollectionID` property.
4. Drag and drop a textbox on the Web form.
5. Set the `TextMode` property to MultiLine.

   NOTE: Set the text box width to at least 400px.

6. On the code-behind page, add the following line.

   ```
   Textbox1.Text = Collection1.XmlDoc.InnerXml
   ```

7. Build the project.
8. View the Web form in a browser.
9. The XML structure of the collection appears in the textbox.

For an additional example, see the Collection XML page on the Developer samples page. It is located at:

- In a browser:

  ```
  http://sitroot/CMS400Developer/Developer/Collection/CollectionXML.aspx
  ```

- In the source code:

  ```
  siteroot/CMS400Developer/Developer/Collection/CollectionXML.aspx
    and CollectionXML.aspx.vb
  ```

Community server controls

8.50 and higher

The Community server controls are as follows:

- *ActivityStream* on the facing page
- *CommunityDocuments* on page 2223
- *CommunityGroupBrowser* on page 2233
- *CommunityGroupList* on page 2237
- *CommunityGroupMembers* on page 2243
- *CommunityGroupProfile* on page 2247
- *ContentFlagging* on page 2251
- *Favorites* on page 2254
- *Friends* on page 2260
- *Invite* on page 2267
- *MessageBoard* on page 2270

ActivityStream

8.50 and higher

The ActivityStream server control displays notification messages generated by Ektron's Notification system. See also: Notifications

When this control is added to a Web page, Ektron looks for a `DefaultObjectID` defined in properties. If one is found, the activity stream is based on that user or group. If none is found, notifications are based on the pages's dynamic query string parameter, which typically identifies the logged-in user.

You can exclude any user or a group from the activity stream. To achieve this, open the page that hosts the Activity Stream server control, find the control, and add the following to the control's code-behind.

```
//activityStream
cmsActivityStream.ExcludeUserIds.Add(this.ProfileId);
You can also exclude groups.
cmsActivityStream.ExcludeGroupIds.Add(GroupId1);
cmsActivityStream.ExcludeGroupIds.Add(GroupId2);
:
cmsActivityStream.ExcludeGroupIds.Add(GroupIdN);
```

To add this control to a page, drop it on a Web form and set the following properties.

- **DefaultObjectID**. If you want a user's or community group's activity stream to appear in the control, enter that ID.
- **DefaultObjectParameter**. Enter the default object parameter used on the QueryString to define an object's ID. For example, if you are passing the ID value of a community group, you might enter 'gid' for this property. So, if you pass `http://~yoursite~/CGHome.aspx?gid=21` in the query string to a page containing this control, you see the activity stream for the community group whose ID is 21.
- **ObjectType**. Select the whether the control is associated with a user or Community Group
- **TemplateUserProfile** and **ProfileParamName**. If you want a user's avatar in the activity stream to be a clickable link that leads to the user's profile page, set these properties.

Inserting the ActivityStream server control onto a page

PREREQUISITE

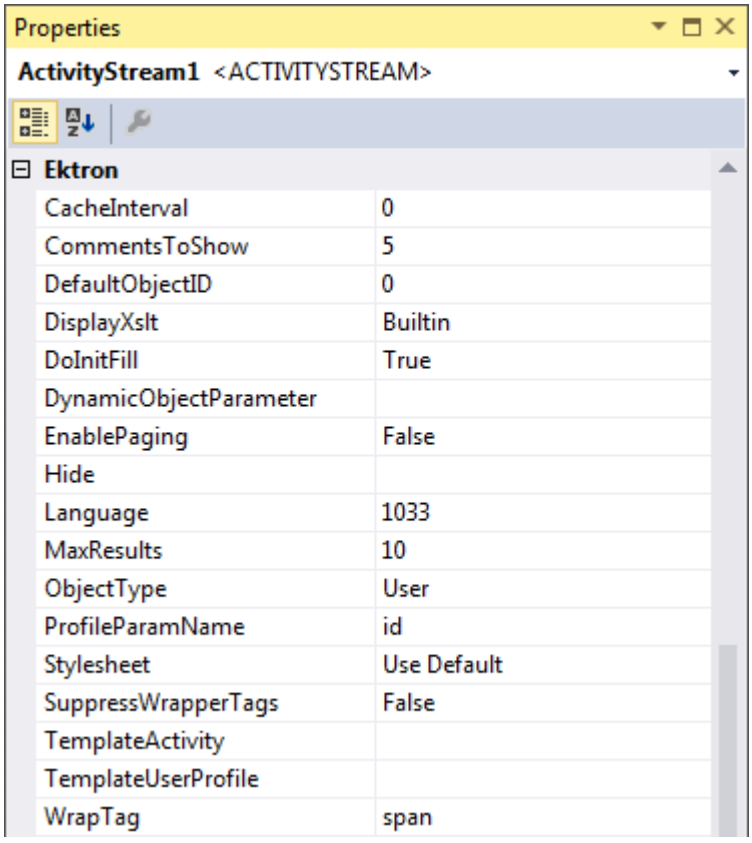
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **ActivityStream** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:ActivityStream ID="ActivityStream1" runat="server" />
```

4. Click on `ActivityStream` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



ActivityStream properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual

Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultObjectID** (Long)

  The ID of the object whose activity stream will appear where you place this server control. For example, if you want this control to display Scott Markey's activities, and his USER ID is 142, place **142** here, and set the `ObjectType` property to **User**. To display the activity stream for the logged-in user, enter zero (**0**).

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicObjectParameter** (String)

  Gets or sets the QueryString parameter to read a object ID dynamically. To use the default object ID, leave blank. For example, if you are passing the ID value of a community group, you might enter 'gid' for this property. So, if you passed `http://~yoursite~/CGHome.aspx?gid=21` on the Query Strings to a page containing this control, you would see the activity stream for the group with an ID of 21.

- **EnablePaging** (Boolean)

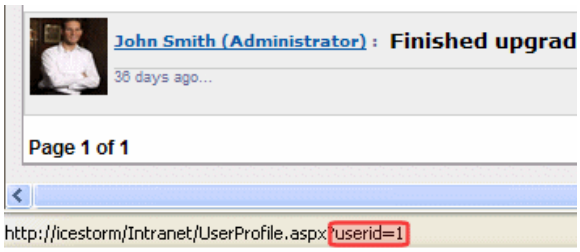
  This property, in conjunction with the **MaxNumber** property, lets site visitors view an unlimited number of items while controlling the amount of screen space. The **MaxNumber** property limits the number of items displayed. If you set this property to **True**, and the number of items exceeds **MaxNumber**, navigation aids appear below the last item, allowing the visitor to go to the next screen.

- **Hide** (Boolean)

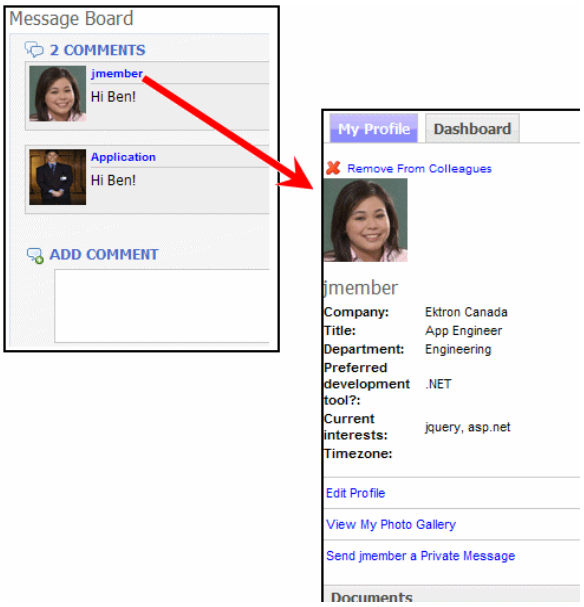
  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **MaxResults** (Integer)
  The Maximum number of notifications to fetch. 0 (zero) = unlimited.
- **ObjectType** (ActivityFeedType)
  The type of object to which this control is assigned. Choices are:
  - **User**. Control is assigned to an individual
  - **Group**. Control is assigned to a community group
- **ProfileParamName** (String)
  The parameter name to pass in the QueryString to the TemplateUserProfile page, if you want it to be anything other than id. For example, you may prefer `userid`, because it is more descriptive, as shown in the following example.



- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TemplateActivity** (String)
  The URL path to a page that contains another ActivityStream server control. When this property contains a path and the destination page has an ActivityStream control whose `ObjectType` property is set to `Activity`, a user can click an Activity Stream's time span on the first page to open a second page that contains just that activity. See also: *Viewing a customer orders list* on page 2394.
- **TemplateUserProfile** (String)
  The URL path to a page that contains the UserProfile server control. The path can be relative or absolute. If you enter a path, a user can click a user's name or avatar from the Message Board server control and be forwarded to the profile page. See illustration.



User templates can be defined in the Ektron **Workarea > Settings > Community Management > Templates** screen. However, if you assign a template in this field, this setting takes precedence over the setting on the Workarea Template screen.

- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

CommunityDocuments

8.50 and higher

The CommunityDocuments server control displays a list of uploaded content filtered by category. In addition, a logged-in member can manage files, create categories, and decide which users can view the documents. This server control is typically placed on a user's or a group profile page.

NOTE: When you drag and drop this control on a Web form in Visual Studio, the following message appears in the label of the control: "Workspace for user does not exist". This happens because no ID is assigned to the control. When you assign an ID, the control's name and the ID are displayed. If you use the `DynamicParameter` property to dynamically pass an ID from the QueryString, the above message appears because an ID is not assigned to the control.

Inserting the CommunityDocuments server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

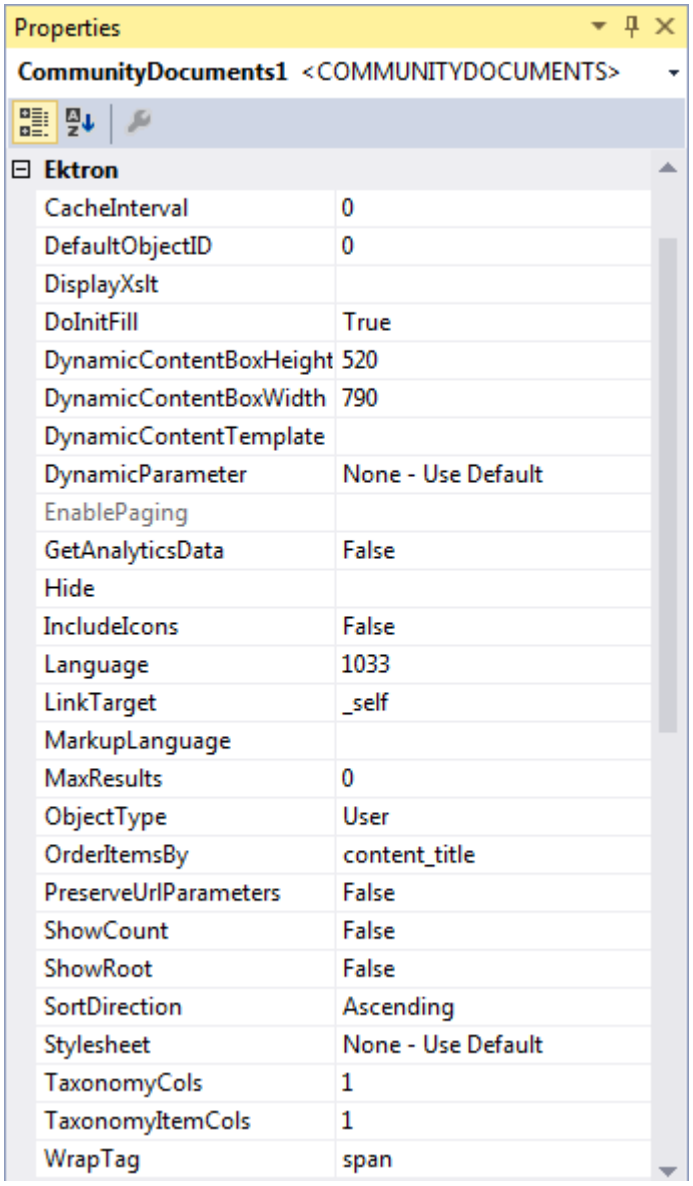
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **CommunityDocuments** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:CommunityDocuments ID="CommunityDocuments1" runat="server" />
   ```

4. Click on `CommunityDocuments` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as

you modify the property values.



CommunityDocuments properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultObjectID** (Long)

  The default object ID for this control to use when there is no matching dynamic parameter value passed.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicContentBoxHeight** (Integer)

  The height of the dynamic content box in pixels.

- **DynamicContentBoxWidth** (Integer)

  The Width of the dynamic content box in pixels.

- **DynamicContentTemplate** (String)

  The template to use when displaying dynamic content. Leave blank to use the dynamic box.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a object ID dynamically. To use the default object ID, leave blank.

- **EnablePaging** (Boolean)

  This property, in conjunction with the **MaxNumber** property, lets site visitors view an unlimited number of items while controlling the amount of screen space. The **MaxNumber** property limits the number of items displayed. If you set this property to **True**, and the number of items exceeds **MaxNumber**, navigation aids appear below the last item, allowing the visitor to go to the next screen.

- **GetAnalyticsData** (Boolean)

  Set this property to **True** if you want the following information for each content in the list. Returns **Content View Count**, **Content Rating**, **Content Rating Average**. Create your own XSLT styles to display this data.

  IMPORTANT: This property provides reliable data only when the Business Analytics Feature is on. See Running Ektron Business Analytics.

- **Hide** (Boolean)

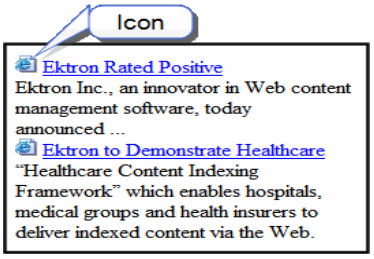
  Hides or displays the output of the control in design time and run time.

- **True**. Hide the control output.
- **False**. Display the control output.

- **IncludeIcons** (Boolean)

  Choose whether to display icons next to the navigation list's links.

  - **True**. Show icons
  - **False**. Hide icons



- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LinkTarget** (String)

  Determines the type of window that appears when you click a link in the server control.

  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.

- **MarkupLanguage** (String)

  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.

  The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.

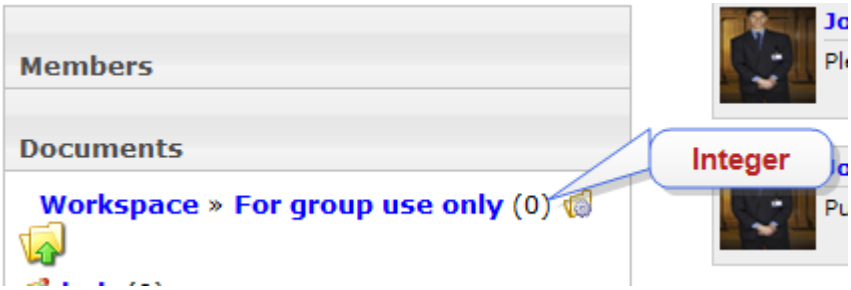
  To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.

  See also: *Ektron Markup Language* on page 2633

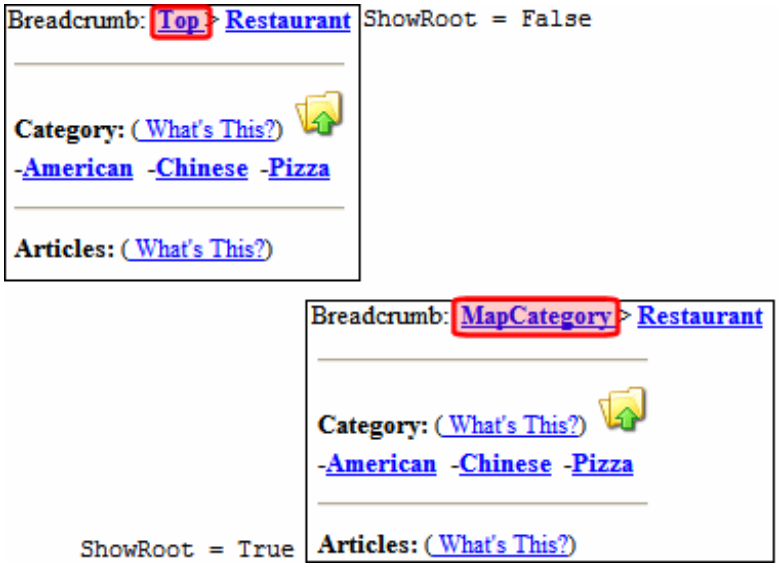
- **MaxResults** (Integer)

  Enter the maximum number of items to appear in the initial display of this server control. To set no maximum, enter zero (0). To let site visitors view more than the maximum but limit the amount of space, enter the maximum number of results per page here. Then, set the **EnablePaging** property to **True**. If you do and more than the number of **MaxResults** are available, navigation aids help the site visitor view additional items.

- **ObjectType** (CommunityDocumentsObjectType)
  The type of object to which this control is assigned. Choices are:
  - **User**. Control assigned to an individual
  - **Group**. Control assigned to a community group
- **ShowCount** (Boolean)
  Determines if an integer representing the number of items in a category appears next to the category.
  - **True**. Show number next to category
  - **False** (default). Do not show number next to category



- **ShowRoot** (Boolean)
  - **False. Top** represents the first node of the taxonomy path.
  - **True**. The *name* of the taxonomy path's first node appears instead of **Top**.



- **SortDirection** (String)
  Select the direction of the `itemSortOrder` property. Choose **Ascending** or **Descending**.
- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder

outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.

- **TaxonomyCols** (Integer)
  Enter the number of columns in which this taxonomy/category appear on the page.
- **TaxonomyItemCols** (Integer)
  Enter the number of columns in which the taxonomy item appears on the page.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Using CommunityDocuments with an individual user profile

Part of the Community Platform, this server control lets users create and upload individual user content to the user profile. The content can be HTML content or an asset, such as a .PDF document or a .jpeg file.

Adding folders to your workspace

1. On the website, navigate to a **User's Profile Page** > **Workspace**.
2. Click **Manage Folder** (). The Add Folder box appears.
3. Enter a **Name** for the folder.
4. Determine with whom you want to share the folder's documents. For a description of the **Share** options, see *Sharing workspace content* on page 2231.
5. Click **Add**. The page refreshes and displays the newly added folder.

Editing a folder name in your workspace

1. On the website, navigate to a **User's Profile Page** > **Workspace**.
2. Click **Edit** (), located to the left of the folder's title. The Edit Folder dialog appears.
3. Change the name of the folder.
4. Click **Save**. The page refreshes and shows the folder's new name.

Deleting a folder from your workspace

NOTE: You can not delete the top-level folder, Workspace.

WARNING! Deleting a folder permanently deletes the assets and HTML content in the folder, and its subfolders.

1. On the website, navigate to a **User's Profile Page** > **Workspace**.
2. Click **Edit** (), located to the left of the folder's title. The Edit Folder dialog appears.
3. Click **Delete**. A dialog box asks you to confirm.
4. Click **OK**.

Adding assets to a workspace

Assets are files that are not HTML content, such as an Office document or PDF.

1. On the website, navigate to a **User's Profile Page** > **Workspace**.
2. Select a folder to which the asset will be added. If you want to create a new folder, see *Adding folders to your workspace* on the previous page
3. Click **Add Asset** (). The Add Asset box appears. Its appearance varies depending on your browser, and so on. To learn about these variations, see Methods for Importing Assets.
4. Import one or more assets, depending on the screen. A status box shows the files being uploaded and then the asset(s) appear in the file list.

Creating HTML content in your workspace

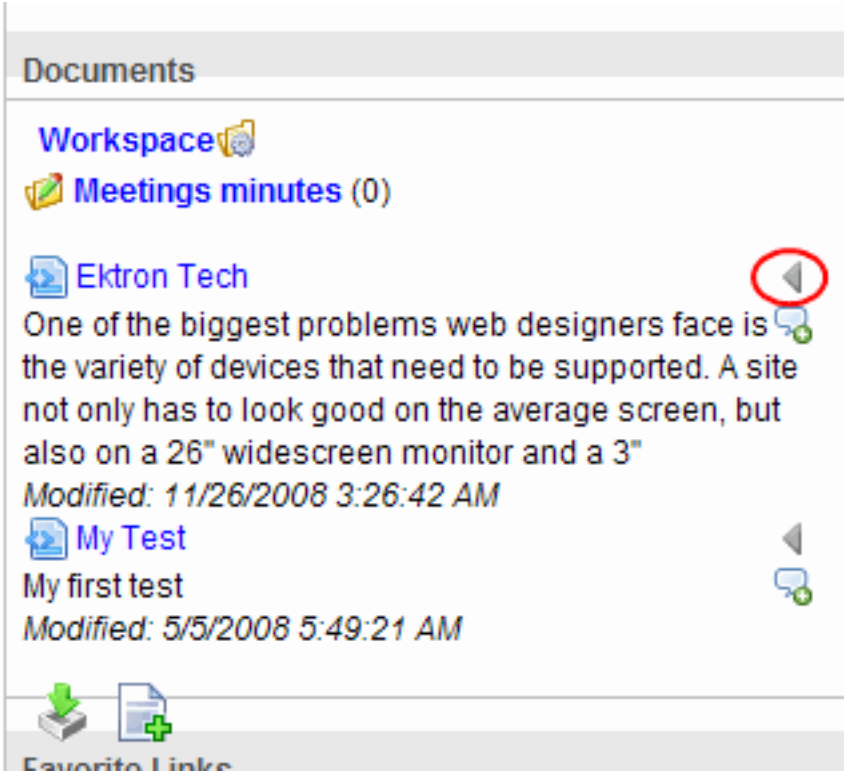
1. On the website, navigate to a **User's Profile Page > Workspace**.
2. Select a folder where the content will be added. If you want to create a new folder, see *Adding folders to your workspace* on the previous page
3. Click **Add HTML Content** (). The Add HTML Content window appears.
4. Add a title and content.
5. Click **Publish** to publish the content.
6. The HTML content appears in the file list.

Moving and copying content in your workspace

You can move or copy assets and HTML content from one folder to another. To accomplish either action:

1. On the website, navigate to a **User's Profile Page** > **Workspace**.
2. Select the folder from which to move or copy the content.

3. Click the drop-down triangle to the right of the content title.



4. From the drop-down list, select **Copy**.
5. Select the folder to which the content will be moved.
6. Click **Manage**.
7. Click **Move Items** ( ) or **Copy** ( ). A dialog box asks you to confirm.
8. Click **OK**. The moved or copied content appears in the folder.

Sharing workspace content

The Workspace area allows users to share content with colleagues. Users can share content with the Public, Colleagues, Selected Colleagues or keep the content private.

Users apply sharing options to *folders*, not individual content items. To share a folder:

1. On the website, navigate to a user's or community group's **Profile Page > Workspace**.
2. Locate the folder you want to share.
3. Click **Manage Folder** ( ). The Add or Share Folder dialog box appears.
4. Click **Share Folder** ( ) in the top right corner. The Share Folder dialog box appears.
5. Select with whom to share your folders.
   - **Public**. Everyone viewing your Workspace
   - **Colleagues**. Only colleagues. See also: *Friends* on page 2260 server control

- **Selected Colleagues**. Only selected colleagues See also: *Designating a selected colleague* on page 2266.
- **Private**. Only you

6. Click **Share**.

Documents in the folder are now available for viewing by the selected viewer type.

Using CommunityDocuments with a group profile

Part of the Community Platform, this server control lets users create and upload group content to the group profile. The content can be HTML content or an asset, such as a .PDF document or a .jpeg file.

Adding folders to the group workspace

Community group members can add folders to the Workspace to help organize items in the **Documents** area.

1. On the website, navigate to a community group's **Profile Page > Workspace**.
2. Click **Manage Folder** (). The Add Folder dialog appears.
3. Enter a **Name** for the folder.
4. Determine with whom you want to share the folder's documents.
   - **Public**. Anyone can view and edit the document
   - **Private**. Only group members can view and edit the document

   NOTE: If you want to later change a folder's share setting, follow the directions in *Sharing workspace content* on the previous page.

5. Click **Add**. The page refreshes and displays the newly added folder.

Editing a folder name in a group workspace

1. On the website, navigate to a community group **Profile Page > Workspace**.
2. Click **Edit** (), located to the left of the folder's title. The Edit Folder dialog appears.
3. Change the name of the folder.
4. Click **Save**. The page refreshes and shows the folder's new name.

Deleting a folder from a group workspace

NOTE: You can not delete the top-level folder, Workspace.

WARNING! Deleting a folder permanently deletes all of the assets and HTML content in the folder, as well as its subfolders.

1. On the website, navigate to a community group's **Profile Page > Workspace**.
2. Click **Edit** (), located to the left of the folder's title. The Edit Folder dialog appears.
3. Click **Delete**. A dialog box asks you to confirm.
4. Click **OK**.

CommunityGroupBrowser

8.50 and higher

The CommunityGroupBrowser server control allows a user to browse a taxonomy structure for community groups. As the user browses, community groups at each level appear in a results box.

In addition, a user can click **Create Group** to create a new group. When the group is created, it is automatically added to the taxonomy you are viewing.

Inserting the CommunityGroupBrowser server control onto a page

PREREQUISITE

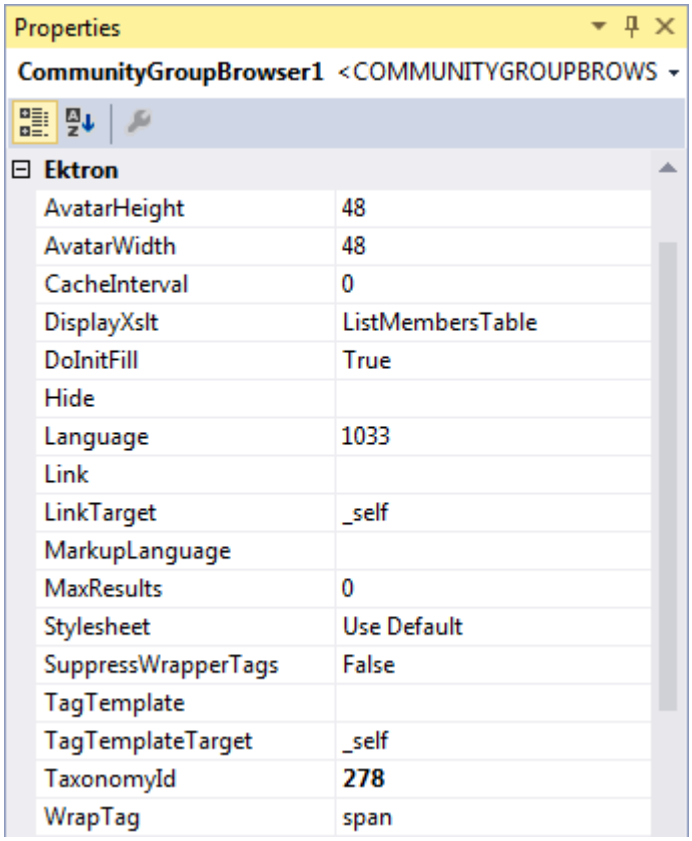
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **CommunityGroupBrowser** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:CommunityGroupBrowser ID="CommunityGroupBrowser1" runat="server" />
```

4. Click on `CommunityGroupBrowser` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



CommunityGroupBrowser properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AvatarHeight** (Integer)

  The display height of the avatar in the results box.

- **AvatarWidth** (Integer)

  The display width of the avatar in the results box.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! If you enter a valid EkML file at the `MarkupLanguage` property, the `DisplayXslt` property value is ignored.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **Link** (String)

  Add a link to the group's profile page. This allows a user to click a link in the community group list and be taken to the group's profile page. There are 2 variables used within the link that represent the group ID and the group name.

  - **{0}**. Represents the group's ID.
  - **{1}**. Represents the group's name.

  You need to have both variables in the link. The Web form can be relative or absolute. Below is an example.

  ```
  groupprofilepage.aspx?gid={0}&gn={1}
  ```

- **LinkTarget** (ItemLinkTargets)

  Determines the type of window that appears when you click a link in the server control.

  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.

- **MarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.

  The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.

  To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.

  See also: *Ektron Markup Language* on page 2633

- **MaxResults** (Integer)
  The Maximum number of items to fetch. 0 (zero) = unlimited.

- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.

- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

- **TagTemplate** (String)
  Add a path to another Web form to create links for the tag text. The path can be relative or absolute. By providing the path to CommunitySearch server control, a user viewing the list of groups can click a tag and search for other groups with the same tag. There are 5 parameters that are automatically added to the link's QueryString that let you pass information about tag.
  - **searchgrptag**. The tag's text for community groups.
  - **TagId**. The tag's ID.
  - **TagLanguage**. The tag's language.
  - **TagCount**. The tag's count (the number of times a tag has been used.)
  - **TagType**. The tag's type: user or community group.

  In addition to these parameters, you can add your own by defining them in the path. If you do, these parameters are appended to yours. See also: Tagging Content, Library Items, Users, and Groups with Keywords.

- **TagTemplateTarget** (String)
  Determines the type of window that appears when you click a link in the server control.

- **_Self** (default). Opens in same window.
- **_Top**. Opens in parent window.
- **_Blank**. Opens in new window.
- **_Parent**. Opens in the parent frame.

- **TaxonomyId** (Long)
  Enter the ID number of the taxonomy or category to appear in this server control. If you don't know the number, click the button and navigate to the taxonomy or category. When you select one, it appears in the center of the Visual Studio window.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

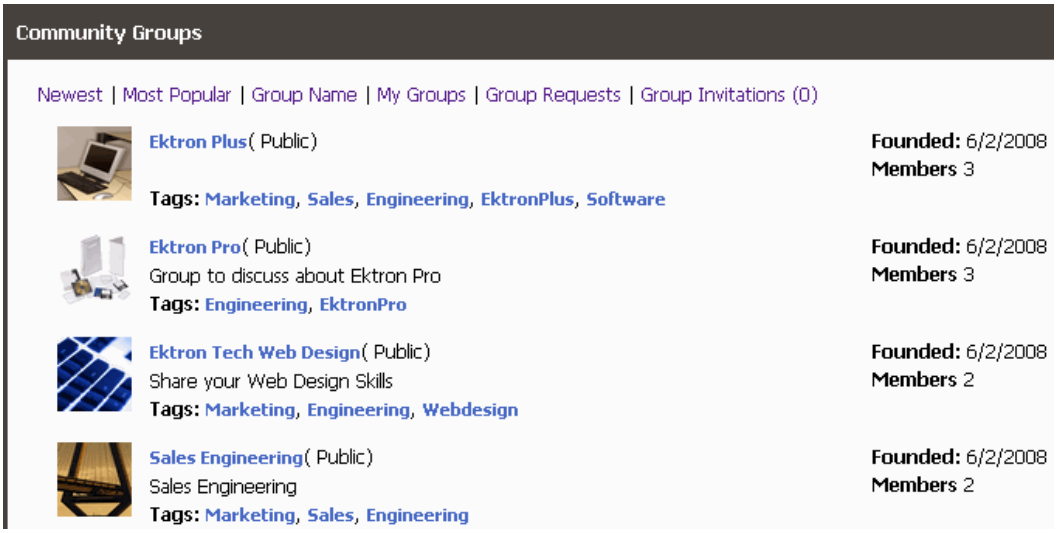
CommunityGroupList

8.50 and higher

The CommunityGroupList server control, part of Ektron's Community Platform, displays a list of community groups. It can be configured to sort groups by name, newest, or most popular.

When associated with a user, this control can show

- community groups with which a user is associated
- community groups the user has been invited to join
- community groups for which the user's request to join is pending



CommunityGroupList server control displays the following information about each community group.

- **Community Group Avatar**. An image representing the group.
- **Community Group Name**. The name of a community group.
- **Type**. Whether the community group is Public or Restricted.
- **Short Description**. The community group's description, as entered in the Short Description field of the group's properties.
- **Tags**. Tags associated with the community group.
- **Founded**. When the community group was created.
- **Members**. The number of users in the community group.

Inserting the CommunityGroupList server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

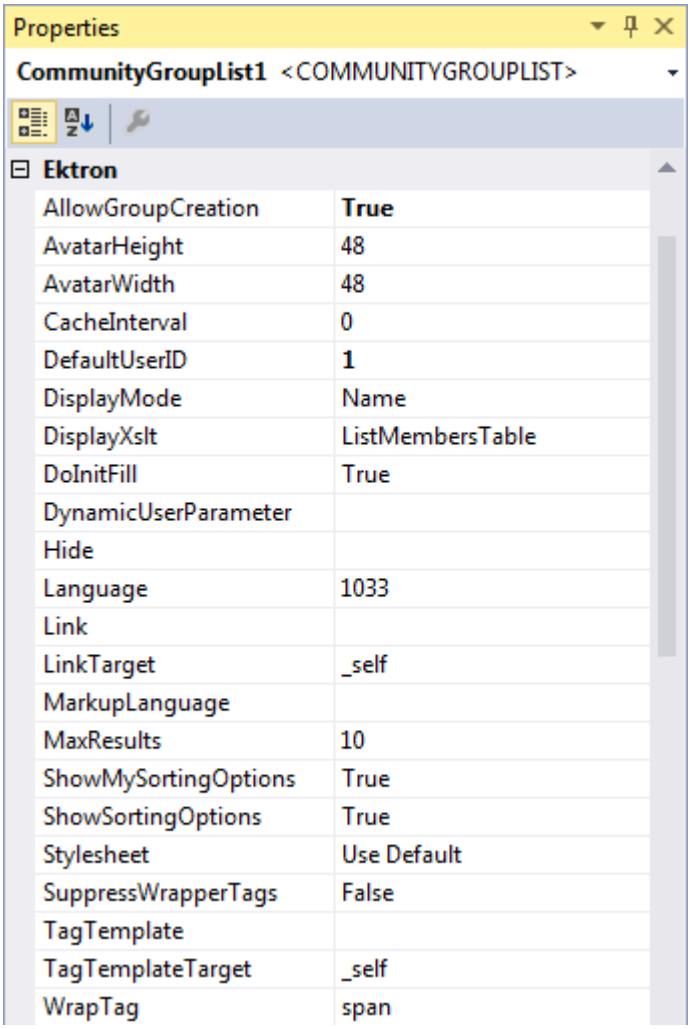
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **CommunityGroupList** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:CommunityGroupList ID="CommunityGroupList1" runat="server" />
   ```

4. Click on `CommunityGroupList` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as

you modify the property values.



CommunityGroupList properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AllowGroupCreation** (Boolean)

  If the user has permission to create community groups and this property is set to True, the Create Groups link appears in the control.

  - **True**. Create Group link appears on the control.
  - **False**. Create Group link is hidden on the control.
- **AvatarHeight** (Integer)

  The display height (in pixels) of the avatar in the results box.
- **AvatarWidth** (Integer)

  The display width (in pixels) of the avatar in the results box.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultUserID** (Long)

  The default User ID for this control to use if no matching dynamic parameter value is passed.

- **DisplayMode** (eDisplayMode)

  Select the way this control initially displays community group information. Choices are:

  - **Newest**. Newly added community groups.
  - **MostPopular**. Community groups with the most members.
  - **Name**. Community groups sorted alphabetically by name
  - **MyGroups**. Community groups to which the logged-in user belongs
  - **MyPendingGroups**. Community groups to which the logged-in user has asked to join, but has not yet been accepted

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING!
  If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicUserParameter** (String)

  Gets or sets the QueryString parameter to read a user ID dynamically. To use the default user ID, leave blank.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **Link** (String)
  Add a link to the group's profile page. This allows a user to click a link in the community group list and be taken to the group's profile page. The link includes these 2 variables.
  - **{0}**. Group ID
  - **{1}**. Group name

  The link requires both variables. The Web form can be relative or absolute. Below is an example.

  ```
  groupprofilepage.aspx?gid={0}&gn={1}
  ```
- **LinkTarget** (ItemLinkTargets)
  Determines the type of window that appears when you click a link in the server control.
  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **MarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.

  The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.

  To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.

  See also: *Ektron Markup Language* on page 2633
- **MaxResults** (Integer)
  The Maximum number of items to fetch. 0 (zero) = unlimited.
- **ShowMySortingOptions** (Boolean)
  If this control is associated with a user and this property is set to True, the controls displays the following sorting options:
  - **My Groups**. A list of community groups to which the user belongs.
  - **Group Requests**. A list of community groups to which a user has requested to join, but has yet to be accepted.
  - **Group Invitations**. A list of community group that the user has been invited to join, but has yet to accept.

When set to False, **Leave Selected Group** is the only option that appears.

Depending on how the `DisplayMode` property is set, you may see additional sorting options.

- **ShowSortingOptions** (Boolean)
  If this control is used to display a general list of community groups and this property is set to `True`, the following sorting options appear.
  - **Newest**. Recently added community groups
  - **Most Popular**. Community groups with the most members.
  - **Group Name**. Community groups sorted alphabetically
- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TagTemplate** (String)
  Add a path to another Web form to create links for the tag text. The path can be relative or absolute. By providing the path to CommunitySearch server control, a user viewing the list of groups can click a tag and search for other groups with the same tag. There are 5 parameters that are automatically added to the link's QueryString that let you pass information about tag.
  - **searchgrptag**. The tag's text for community groups.
  - **TagId**. The tag's ID.
  - **TagLanguage**. The tag's language.
  - **TagCount**. The tag's count. The tag's count is the amount of times a tag has been used.
  - **TagType**. The tag's type. The tag's type will be user or community group.

  In addition to these parameters, you can add your own by defining them in the path. When you do, these parameters will be appended to your parameters.
- **TagTemplateTarget** (ItemLinkTargets)
  Determines the type of window that appears when you click a link in the server control.
  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.

- **_Blank**. Opens in new window.
- **_Parent**. Opens in the parent frame.

- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

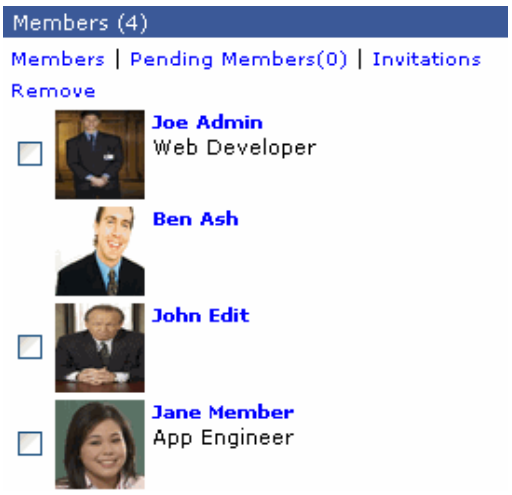
CommunityGroupMembers

8.50 and higher

The CommunityGroupMembers server control allows you display a list of members

- in a community group
- pending approval to join the group
- invited to join the group

In addition, if a logged-in user is the community group's administrator, assigned the Community Group Administrator role, or a site administrator, the user can approve pending members, remove members, and cancel invitations.



The following links appear on the CommunityGroupMembers server control when a community group's administrator or Ektron administrator is logged in and viewing the control.

- **Members**. Current members.
- **Pending Members**. Members who have asked to join the group, but have yet to be accepted. This link is used when access to the community group is restricted.
- **Invitations**. Members who were invited to join the group but have not accepted. This link is used when access to the community group is restricted.
- **Remove**. Removes member from community group.

Inserting the CommunityGroupMembers server control onto a page

PREREQUISITE

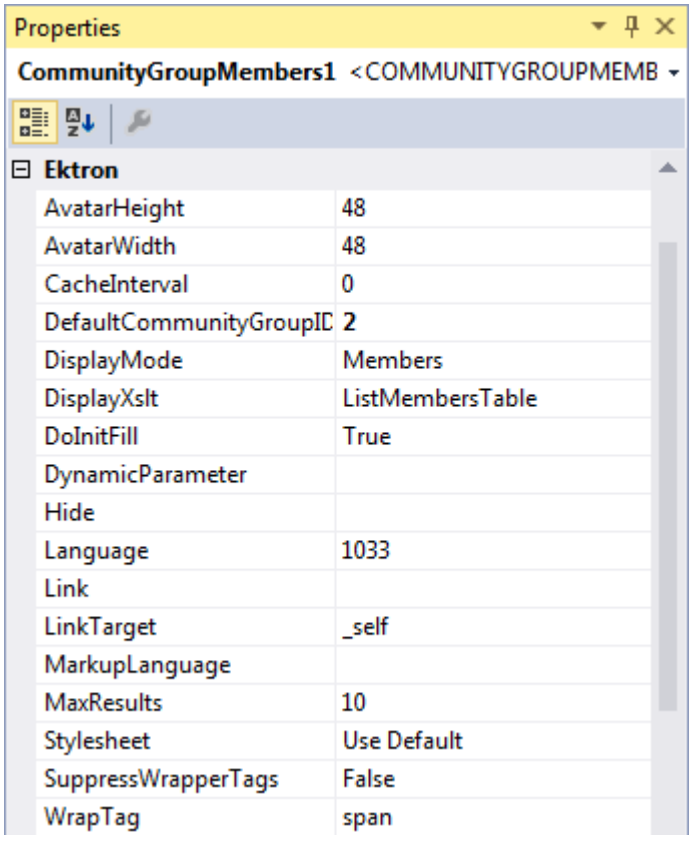
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **CommunityGroupMembers** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:CommunityGroupMembers ID="CommunityGroupMembers1" runat="server" />
```

4. Click on `CommunityGroupMembers` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



CommunityGroupMembers properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual

Studio® help.

- **AvatarHeight** (Integer)
  The display height (in pixels) of the avatar in the results box.
- **AvatarWidth** (Integer)
  The display width (in pixels) of the avatar in the results box.
- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **DefaultCommunityGroupID** (Long)
  The default community group ID for this control to use if no matching dynamic parameter value is passed.
- **DisplayMode** (eDisplayMode)
  Select whether this control displays a community group's current members or their pending members. Choices are:
  - **Members**. Current group members
  - **PendingMembers**. Users who asked or were invited to join the group
- **DisplayXslt** (String)
  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

  WARNING! If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)
  Gets or sets the QueryString parameter to read a community group ID dynamically. To use the default community group ID, leave blank.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **Link** (String)
  Add a link to the member profile page's Web form. This allows a user to click a link in the community group members list and be taken to a member's profile page. Two 2 variables used within the link represent the user's ID and name.
  - **{0}**. User ID
  - **{1}**. User name

  The link needs both variables. The Web form can be relative or absolute. For example:

  ```
  userprofilepage.aspx?gid={0}&gn={1}
  ```
- **LinkTarget** (ItemLinkTargets)
  Determines the type of window that appears when you click a link in the server control.
  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **MarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.

  The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.

  To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.

  See also: *Ektron Markup Language* on page 2633
- **MaxResults** (Integer)
  The maximum number of items to fetch. 0 (zero) = unlimited.
- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.

- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

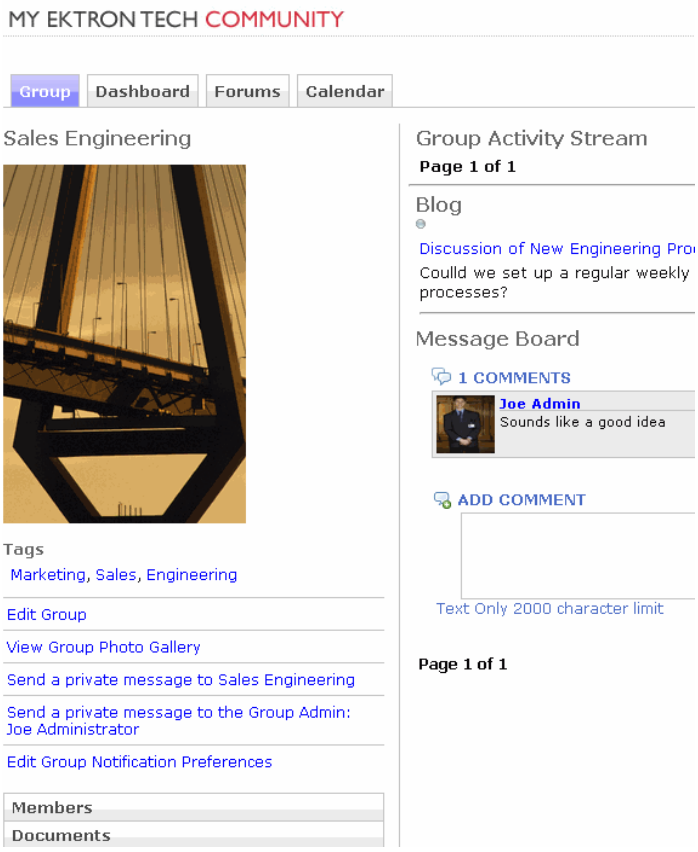
CommunityGroupProfile

8.50 and higher

The CommunityGroupProfile server control displays the profile of a community group. You can see an example of the CommunityGroupProfile server control in the **Ektron Tech** site **> Community > Community Groups >** select any group. The profile includes:

- an image associated with the group
- a description of the group
- tags assigned to the group
- whether the group is Public or Restricted
- the date the group was founded
- location information
- group members
- group administrator

- an **Edit** link that lets you update profile information and group settings



Inserting the CommunityGroupProfile server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

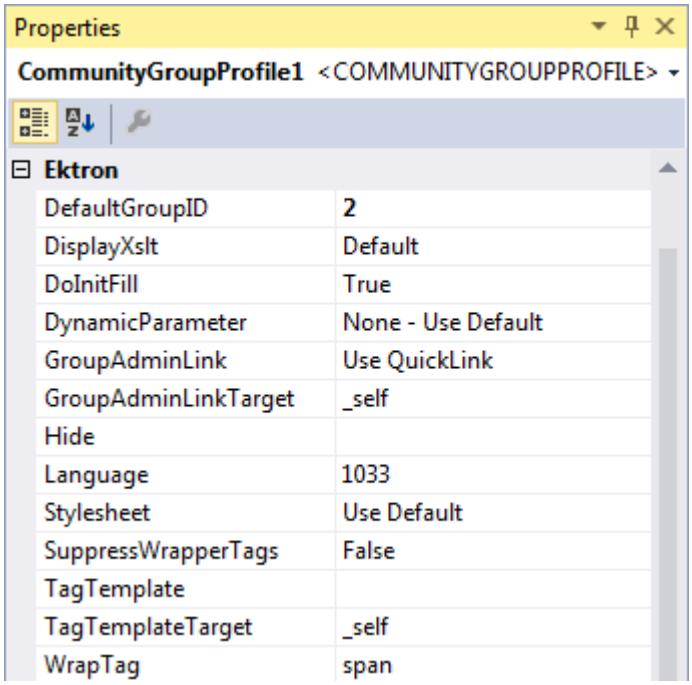
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **CommunityGroupProfile** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:CommunityGroupProfile ID="CommunityGroupProfile1" runat="server" />
```

4. Click on `CommunityGroupProfile` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as

you modify the property values.



CommunityGroupProfile properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **DefaultGroupID** (Long)

  The default community group ID for this control to use when no matching dynamic parameter value is passed.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

  WARNING! If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a community group ID dynamically. To use the default community group ID, leave blank.

- **GroupAdminLink** (String)

  Add a link to the administrator's profile page. This lets a user click a link in the community group page's profile area and be directed to the administrator's profile page. Two variables in the link identify the administrator.

  - **{0}**. Administrator's ID
  - **{1}**. Administrator's name

  Both variables must be in the link. The Web form can be relative or absolute. Below is an example.

  ```
  adminprofilepage.aspx?aid={0}&an={1}
  ```

- **GroupAdminLinkTarget** (ItemLinkTargets)

  Determines the type of window that appears when you click a link in the server control.

  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

- **TagTemplate** (String)
  Add a path to another Web form to create links for the tag text. The path can be relative or absolute. By providing the path to CommunitySearch server control, a user viewing the profile can click a tag and search for other groups with the same tag. Five parameters, which are automatically added to the link's QueryString, let you pass information about the tag.
  - **searchgrptag**. The tag's text for community groups
  - **TagId**. The tag's ID
  - **TagLanguage**. The tag's language
  - **TagCount**. The tag's count. That is, the number of times a tag has been used.
  - **TagType**. The tag's type: user or community group

  You can also add your own parameters by defining them in the path. If you do, these parameters will be appended to yours. See also: Tagging Content, Library Items, Users, and Groups with Keywords
- **TagTemplateTarget** (ItemLinkTargets)
  Determines the type of window that appears when you click a link in the server control.
  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

ContentFlagging

8.50 and higher

Flagging lets site visitors provide feedback about a piece of content, thereby enabling site visitors to moderate your site. This is especially important for sites with large amounts of visitor-generated content. Flagging also helps site visitors feel invested in the site.

Inserting the ContentFlagging server control onto a page

PREREQUISITE

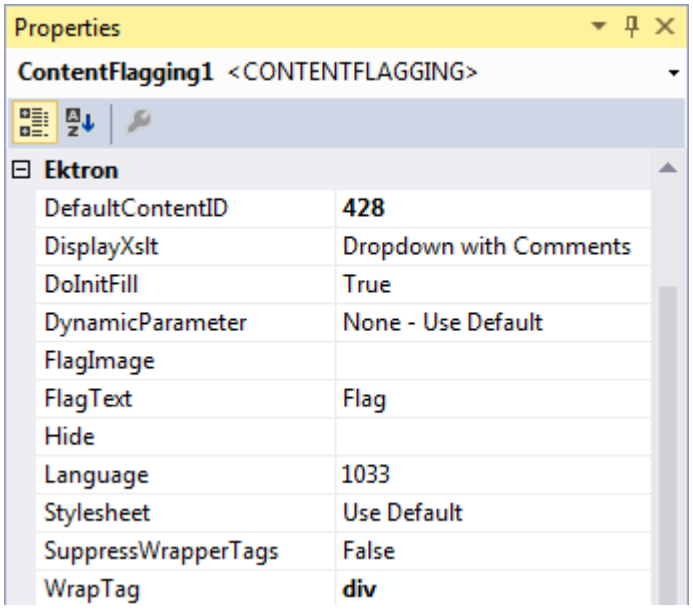
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **ContentFlagging** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:ContentFlagging ID="ContentFlagging1" runat="server" />
```

4. Click on `ContentFlagging` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



ContentFlagging properties

The ContentFlagging server control has the following properties. See also: Defining Flags for Content

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **DefaultContentID** (Long)

  The default content ID for this control to use when there is no matching dynamic parameter value passed.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a content ID dynamically. To use the default content ID, leave blank.

- **FlagImage** (String)

  Enter a path to an image that overrides the text in the `FlagText` property. When the image is clicked, the flag dialog appears.

- **FlagText** (String)

  Text for the link that is shown to allow flagging. For example, you might use "Click here to flag this content." The `FlagImage` property overrides this property. When the text link is clicked, the flag dialog appears.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

- **WrapTag** (String)

  Lets a developer specify a server control's tag.

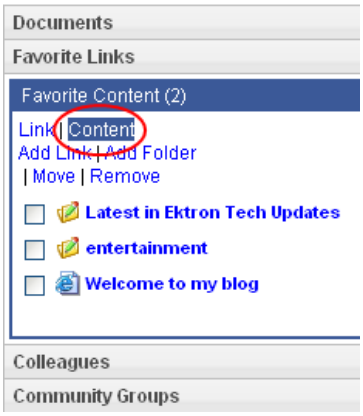
  - **Span** (default). Designate an inline portion of an HTML document as a span element.

- **Div**. Apply attributes to a block of code.
- **Custom**. Lets you use a custom tag.

Favorites

8.50 and higher

The Favorites server control displays a list of Ektron content and external Web links (URLs) that a user has designated as favorite content. Within the control, Ektron content and URLs appear in separate lists. To view favorite content, click **Content**. To view a list of external URLs, click **Link**.



To learn about designating Ektron *content* as a favorite, see *SocialBar* on page 2312.

Inserting the Favorites server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

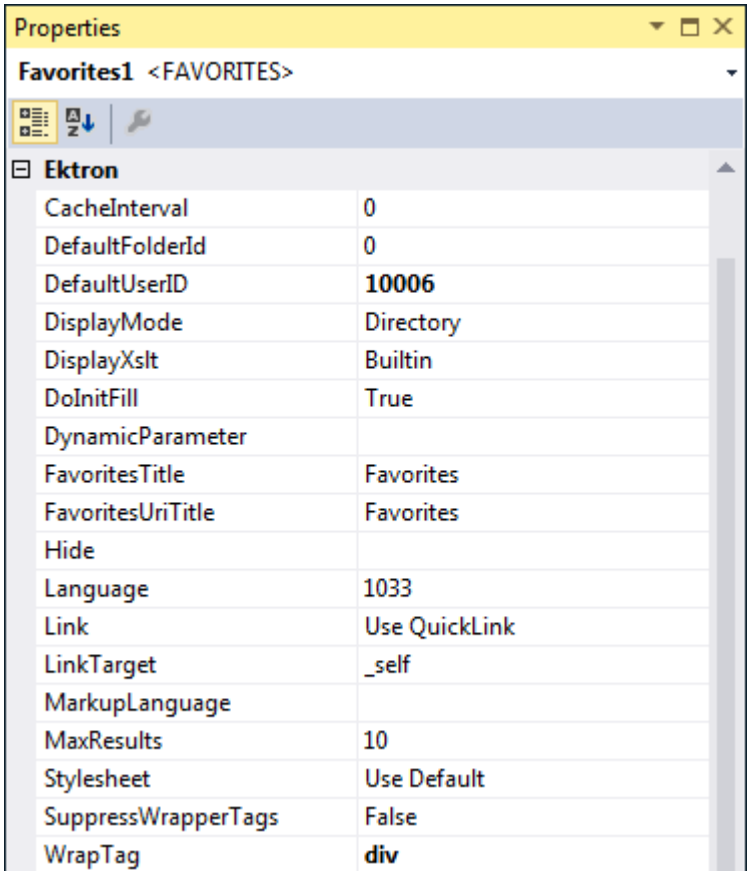
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Favorites** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:Favorites ID="Favorites1" runat="server" />
   ```

4. Click on `Favorites` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



Favorites properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **DefaultFolderID** (Long)
  The default folder ID for this control to use when there is no matching dynamic parameter value passed.
- **DefaultUserID** (Long)
  The default user ID for this control to use when there is no matching dynamic parameter value passed.
- **DisplayMode** (eDisplayMode)
  Select the way this control displays community group information. Choices are:

- **List**. Shows a list of Favorites sorted in alphabetical order.
- **Directory**. Groups favorites by folder. If this selection is chosen, the following menu items are added to the control:
- **Add Folder**. Allows a user to add a sub folder.
- **Move**. Allows a user to move content to a folder. This item only appears if at least one folder exists. See also: *Grouping favorites by folder* on page 2258

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

  WARNING!
  If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.

- **DoInitFill** (Boolean)

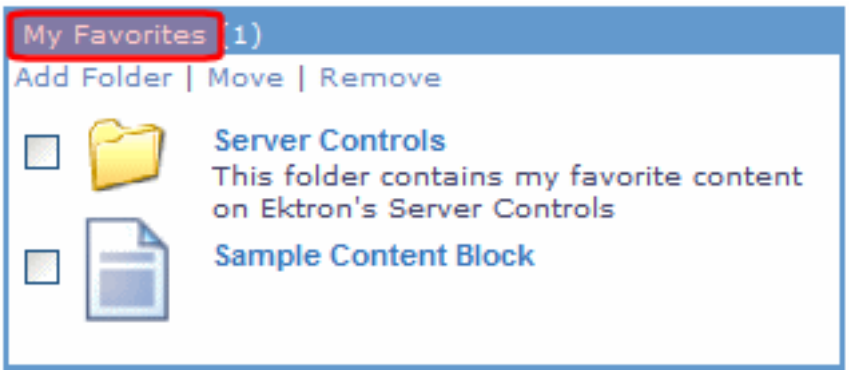
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a user ID dynamically. To use the default user ID, leave blank.

- **FavoritesTitle** (String)

  The title shown at the top of the control.



- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)(missing snippet link)
- **Link** (String)

  Enter a link to the content Web page. This allows a user to click a link in the Favorites control and be taken to the content.

  - **{0}**. Content ID
  - **{1}**. Content language
  - **{2}**. Content title

  The link must have all 3 variables. The Web form can be relative or absolute. Below is an example.

  ```
  contentpage.aspx?cid={0}&lang={1}&title={2}
  ```

- **LinkTarget** (ItemLinkTargets)

  Determines the type of window that appears when you click a link in the server control.

  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **MarkupLanguage** (String)

  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.See also: *Ektron Markup Language* on page 2633*Ektron Markup Language* on page 2633

- **MaxResults** (Integer)

  The Maximum number of items to fetch. 0 (zero) = unlimited.

- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Adding a URL to your favorites

1. Log in and navigate to the profile page that hosts the Favorites control.
2. Click **Add Link**.
3. In the **Name** field, add a name for the URL link.
4. In the **Link** field, add the URL.
5. In the **Description** text area, add a description for the URL. This is optional.
6. Click **OK**.

After a URL is added, it appears on the control's **Links** list.

You also can add a URL to your Favorites by clicking the **Add to Favorites** link on the Social Bar server control. See .

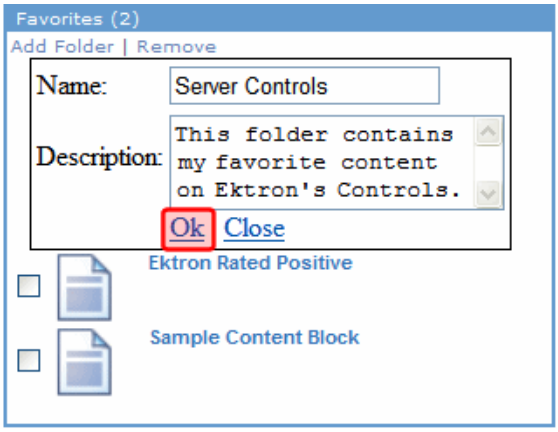
Grouping favorites by folder

The `DisplayMode` property controls whether a user's Favorites are displayed as a basic list, or if they can be organized in folders. For example, a user groups all content created by a certain author in a folder with that author's name. If you add a folder to a user's favorites, content in the folder appear in the **Content** list; URLs in the folder appear in the **Links** list.

To move a Favorite to a folder:

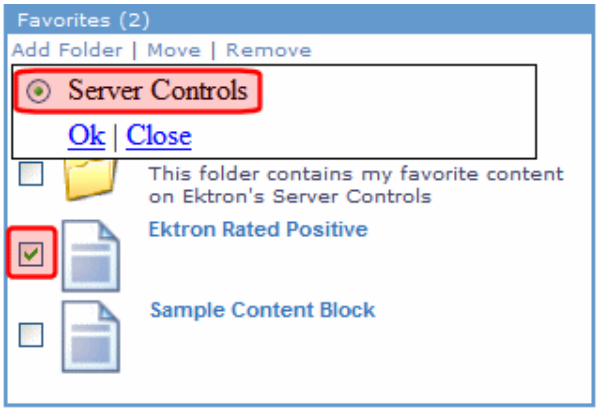
1. Make sure the Favorites server control's `DisplayMode` property is set to **Directory**.
2. Log in to the website.
3. Navigate to your profile page or the page the contains the Favorites server control.
4. In the Favorites server control, click **Add Folder**.
5. Enter a **Name** and **Description** for the folder.

6. Optionally check one or more favorite items then click **OK**.



7. Click **Move** on the Favorites server control.

8. Select a folder to which the items will be moved.



9. Click **OK**. A dialog box appears asking you confirm moving the items. You can find the Favorite content in the folder. Click the folder to see all content in that folder.

Deleting a favorites folder

WARNING! Deleting a folder also deletes all content links and URL links in the folder. If you want to retain the links, move them out of the folder before deleting it. To remove a link from a folder, click the folder, check the link, and click **MoveUp One Level**. Deleting a folder also deletes all of its subfolders.

1. Navigate to your profile page or a page the contains a Favorites server control.
2. In the **Favorites** section of the screen, click **Edit Folder** ( ) next to the folder's title. A dialog box appears.
3. Click **Delete**. A confirmation box appears.
4. Click **OK**. The folder is deleted.

Friends

8.50 and higher

The Friends server control typically appears on your profile page and displays your friends (colleagues). If you browse to another user's profile, you see that user's colleagues.

Inserting the Friends server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

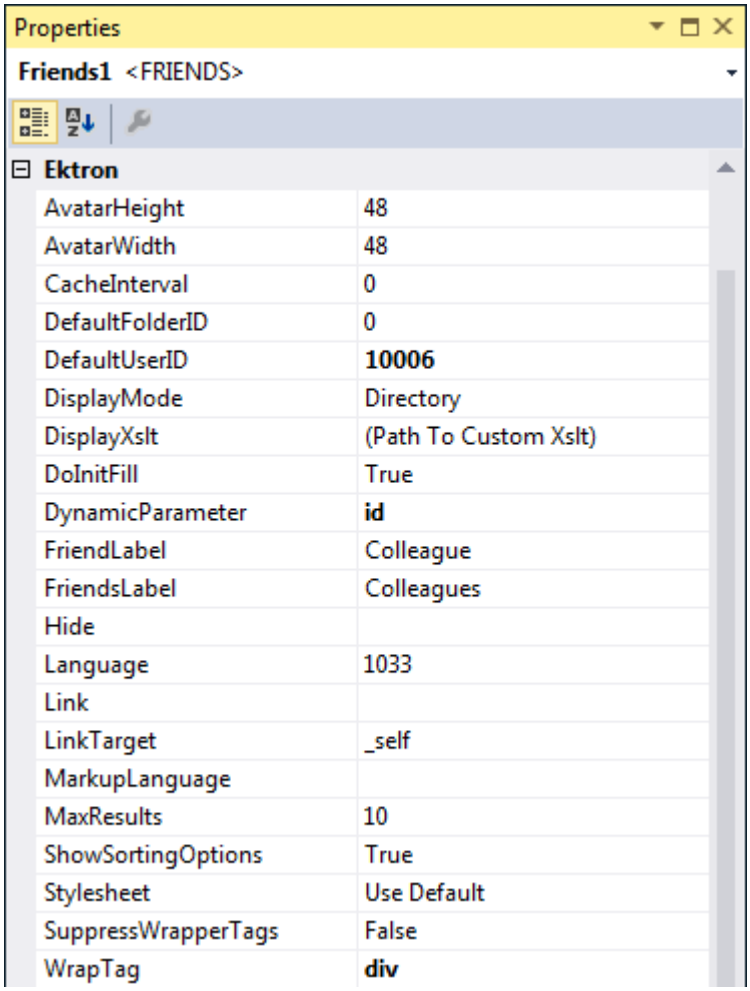
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Friends** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Friends ID="Friends1" runat="server" />
```

4. Click on `Friends` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the

property values.



Friends properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AvatarHeight** (Integer)

  The display height in pixels of the avatar in the results box. 0 (zero) = unlimited.

- **AvatarWidth** (Integer)

  The display width in pixels of the avatar in the results box. 0 (zero) = unlimited.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultFolderID** (Long)

  The folder ID that contains the user ID to display. If a DefaultContentID is given, it overrides this property.

- **DefaultUserID** (Long)
  The default user ID for this control if no matching dynamic parameter value is passed.
- **DisplayMode** (eDisplayMode)
  Select the way this control displays colleagues. Choices are:
  - **Directory**. Allows users to group colleagues by folders. In this mode, these menu items appear.
  - **Add Folder**. Lets user add a folder.
  - **Move**. Lets a user place colleagues in a folder. This option appears only if a folder exists.
  - **List**. Lists colleagues in alphabetical order
  - **Pending**. Lists users who have sent colleague requests to the logged-in user. The user can approve or decline these requests.
  - **SentInvites**. Lists users to whom the logged-in user sent colleague requests. The user can cancel invitations that have not been accepted yet.
- **DisplayXslt** (String)
  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

  WARNING!
  If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)
  Gets or sets the QueryString parameter to read a user ID dynamically. To use the default user ID, leave blank.
- **FriendLabel** (String)
  Set the singular text to be used as a title for the Friends server control. The default value is **Colleague**.
- **FriendsLabel** (String)
  Set the plural text to be used as a title for the Friends server control. The default value is **Colleagues**.

- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **Link** (String)
  Enter a link to the user profile page template. The path can be relative or absolute. This value is referenced when a user clicks another user in the Friends control. Upon clicking the link, the selected user's profile page appears. The link requires these variables.
  - **{0}**. Represents user ID
  - **{1}**. Represents user display name

  For example: `userprofilepage.aspx?uid={0}&dn={1}`
- **LinkTarget** (ItemLinkTargets)
  Determines the type of window that appears when you click a link in the server control.
  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **MarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.See also: *Ektron Markup Language* on page 2633
- **MaxResults** (Integer)
  The maximum number of colleagues to display in the control. 0 (zero) = unlimited.
- **ShowSortingOptions** (Boolean)
  Determines if sorting options appear on the control. If set to **True**, the following options appear:
  - **Colleagues**. All current colleagues
  - **Pending**. Colleagues who sent you a colleague request that you have not

yet accepted
    - **Invited**. Colleagues to whom you have sent colleague requests
- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.

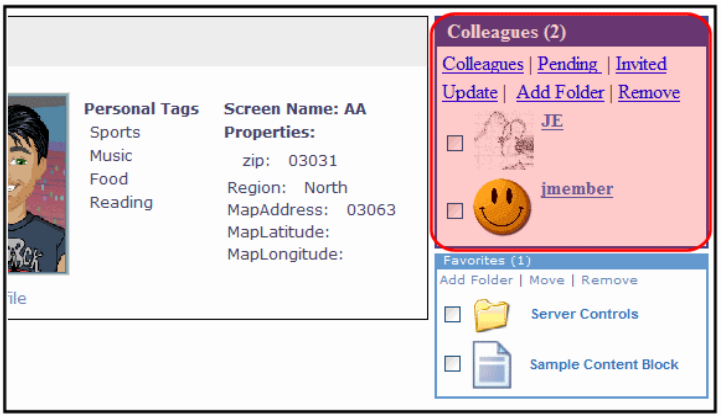
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
    - **True**. Suppress wrap tags.
    - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
    - **Span** (default). Designate an inline portion of an HTML document as a span element.
    - **Div**. Apply attributes to a block of code.
    - **Custom**. Lets you use a custom tag.

Using the Friends server control

The Friends server control displays a list of:

- **current colleagues**. Users who accepted your invitation to be a colleague, or whose invitation to be a colleague you accepted
- **pending colleagues**. Users who sent you a colleague request, which you have not yet accepted
- **invited colleagues**. Users to whom you have sent a colleague requests, which has not yet been accepted

The control can lets you navigate between views.



To manage existing colleagues on the website, find the **Colleague** area of your profile page. Here you can view, approve, and remove colleagues.

Managing colleagues

The following sections explain how to manage colleagues.

Viewing pending colleague requests

Pending colleagues are users who sent you a colleague request which you have not yet accepted. To view pending colleagues on the website, navigate to the **Pending Colleagues** area of your profile page. From here, you can accept or decline colleague requests.

Accepting a colleague request

Accepting a colleague request adds the user to your colleagues list. This lets you access additional information on the colleague's profile page.

1. Navigate to the **Pending colleagues** area of your profile page.
2. Place a check in the box next to each colleague whose request you want to accept.
3. Click **Approve**. The page refreshes, and the selected users are removed from the Pending Colleagues list and added to the colleagues list.

Declining a colleague request

Declining a colleague request cancels the request. When you decline the request, it is removed from your Pending Colleagues list and the requester's Invited list.

1. Go to the **Pending Colleagues** area of your profile page.
2. Place a check in the box next to each colleague whose request you want to decline.
3. Click **Remove Selected**. A dialog asks you to confirm.
4. Click **OK**. The page refreshes, and selected users are removed from your Pending Colleagues list and requester's Invited list.

Viewing invited colleagues

The Invited colleagues list displays users to whom you have sent colleague invites. From this screen, you can delete requests that have not yet been accepted. See also: *Viewing invited colleagues* above To view sent colleague requests, go to the **Invited Colleagues** area of your profile page.

Canceling invited colleagues

1. Go to the **Invited Colleagues** area of your profile page.
2. Place a check in the box next to each colleague whose invite you want to cancel.
3. Click **Remove Selected**.

Removing colleagues

1. Go to the **Colleagues** area of your profile page.
2. Place a check in the box next to each colleague you want to remove.
3. Click **Remove Selected**.

Designating a selected colleague

Selected colleague is a special category of user who can be given access to documents and photos that regular colleagues cannot view. A user makes this designation when sharing a Workspace, which can be shared with the Public, all colleagues, or selected colleagues.

1. Go to the **Colleagues** area of your profile page.
2. Check the box next to each colleague you want to make a Selected Colleague.
3. Click **Update**. **Selected Colleague** appears next to the colleague.

To change a selected colleague back to a colleague, check the box next to a selected colleague and click **Update**.

Creating a colleague group folder

Grouping lets you organize colleagues by folder. For example, you place all family members in a "Family" folder. You can only group current colleagues, not colleagues who are pending or invited.

The following sections explain how to work with Colleague folders.

Creating a colleague group folder

1. Make sure the `DisplayMode` property of the Friends server control is set to **Directory**.
2. Log in to the website.
3. Navigate to your profile page or a page the contains a Friends server control.

4. Click **Add Folder** in the Friends server control.
5. Enter a **Name** and **Description** for the folder.
6. Click **OK**.

Placing a colleague in a group folder

1. Navigate to your profile page or a page the contains a Friends server control.
2. Check one or more colleagues.
3. Click **Move** on the Friends server control. A list of folders appears. Select the folder to which you want to move them.
4. Click **OK**.

Renaming a folder

1. Navigate to your profile page or a page the contains a Friends server control.
2. In the **Colleagues** section of the screen, click **Edit Folder** () next to the folder's title. A dialog box appears.
3. Change the name, description or both.
4. Click **Save**.

Deleting a folder

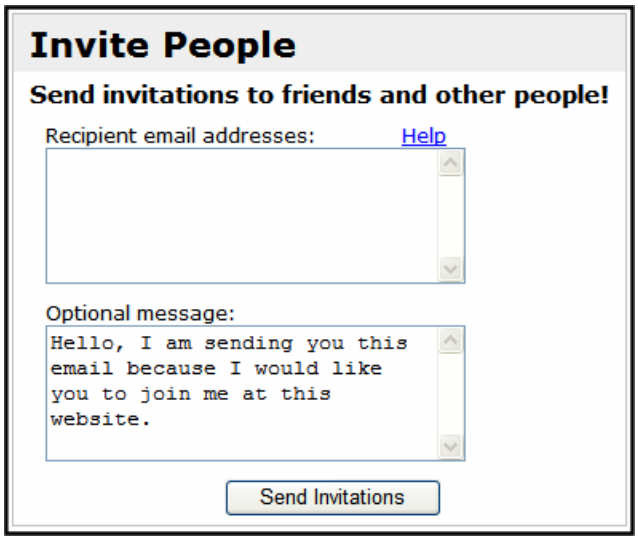
WARNING! Deleting a folder deletes all colleagues within it. To retain the colleagues, move them from the folder first by clicking the folder, checking the colleague, and clicking **Move Up One Level**.
Deleting a folder also deletes all of its subfolders.

1. Navigate to your profile page or a page the contains a Friends server control.
2. In the **Colleagues** section of the screen, click **Edit Folder** () next to the folder's title. A dialog box appears.
3. Click **Delete**.
4. Click **Save**.

Invite

8.50 and higher

The Invite server control lets a site visitor invite people to join a site or become a colleague. The control displays a dialog box that prompts for several email addresses. The dialog includes an optional message, which appears in the body of the email. The site visitor can edit or delete the message.



NOTE: Messages are defined in the Workarea. For more information, see Generating email Invitations for Community Management.

Invitations have a single "from" email address. Its default value is `invitations@example.com`. Your site administrator should change it to one that is appropriate for your organization. To change the "from" email address for invitations, open your site's `web.config` file and change the value of following key.

```
<add key="ek_InvitationFromEmail" value="invitations@example.com"/>
```

WARNING! The default *From* email address used to send all invitations is invitation@example.com. Episerver strongly recommends changing this address. See also: Generating email Invitations for Community Management.

Inserting the Invite server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

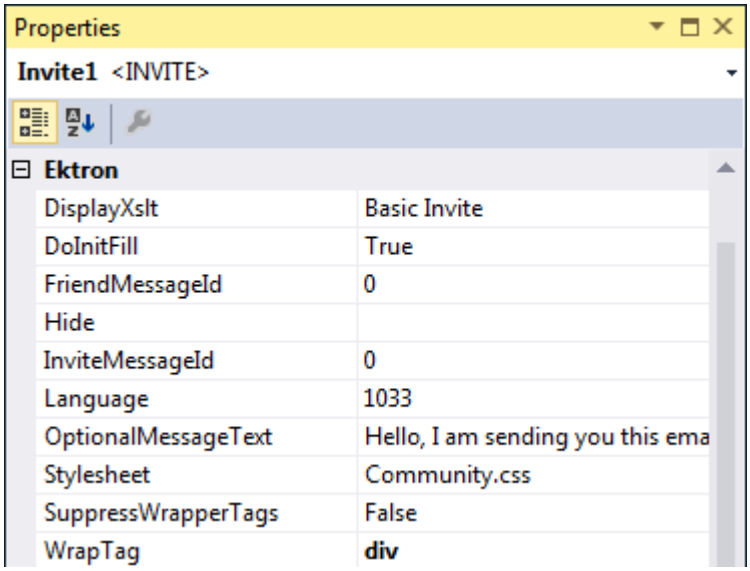
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Invite** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:Invite ID="Invite1" runat="server" />
   ```

4. Click on `Invite` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the

property values.



Invite properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

  WARNING!
  If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **FriendMessageId** (Long)

  The ID of the message to send in the link section of a FriendInvitation type email. This message is defined in the **Workarea > Settings > Community Management > Messages**. If set to 0 (zero), the server control uses the default message.

- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **InviteMessageId** (String)
  The ID of the message to send in the link section of a GroupInvitation or FriendInvitation email. This message is defined in the **Workarea > Settings > Community Management > Messages**. If set to 0 (zero), the server control uses the default message.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **OptionalMessageText** (String)
  The body text of the email message. The text appears in the Invite server control's dialog box and can be edited by site visitors when they use the Invite control.
- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

MessageBoard

8.50 and higher

The Message Board server control allows a user to submit comments on a Web page about a user, community group, or content item.

Comments are presented in a "wall" type format, which means that the newest one appears at the top, and older comments are pushed down as new ones arrive.

Other users can reply to comments. Replying to Message Board comments facilitates community building by allowing members to share information and, thereby, feel they have a personal stake in the discussion.

Comments are posted to the Message Board immediately unless the board is moderated. See also: Moderating a Message Board.

On a PageBuilder page, you can insert a Message Board using the Message Board widget.

Inserting the MessageBoard server control onto a page

PREREQUISITE

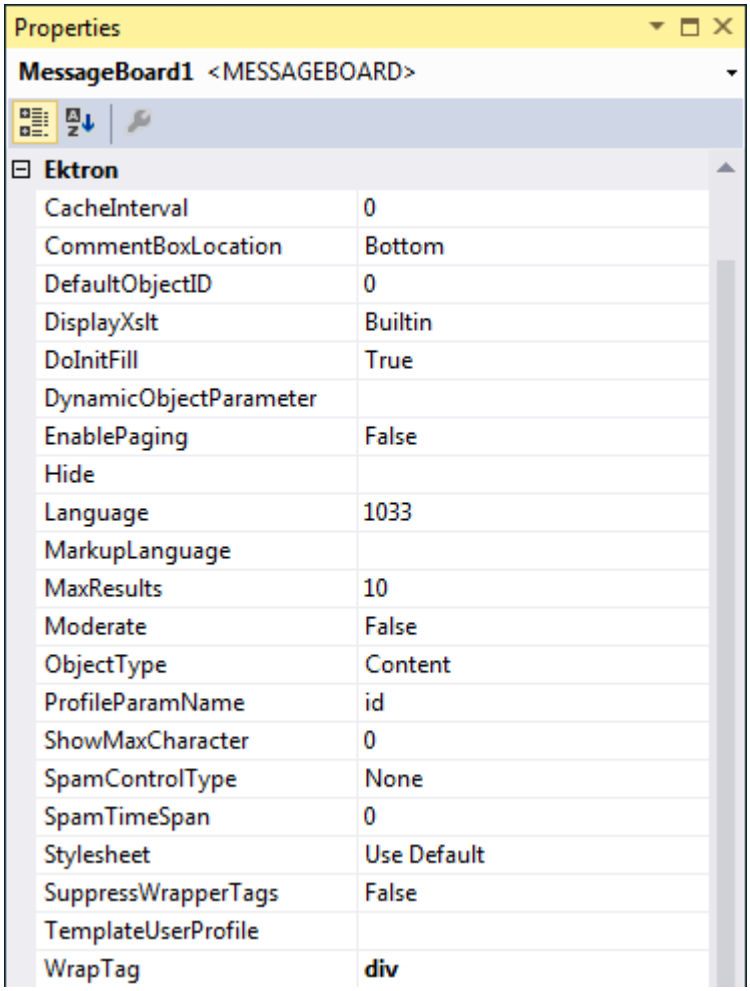
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **MessageBoard** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:MessageBoard ID="MessageBoard1" runat="server" />
```

4. Click on `MessageBoard` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



MessageBoard properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **CommentBoxLocation**

  Defines the location of the comment box. By default, bottom is selected. Choices are:

  - **Bottom**. Comment box appears below the comments
  - **Top**. Comment box appears above the comments
  - **Data Type:** EkEnumeration.MessageBoardCommentBoxLocation
- **DefaultObjectID** (Long)

  The default object ID for this control to use when there is no matching dynamic parameter value passed.
- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

  WARNING!
  If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.
- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicObjectParameter** (String)

  Gets or sets the QueryString parameter to read a object ID dynamically. To use the default object ID, leave blank.
- **EnablePaging** (Boolean)

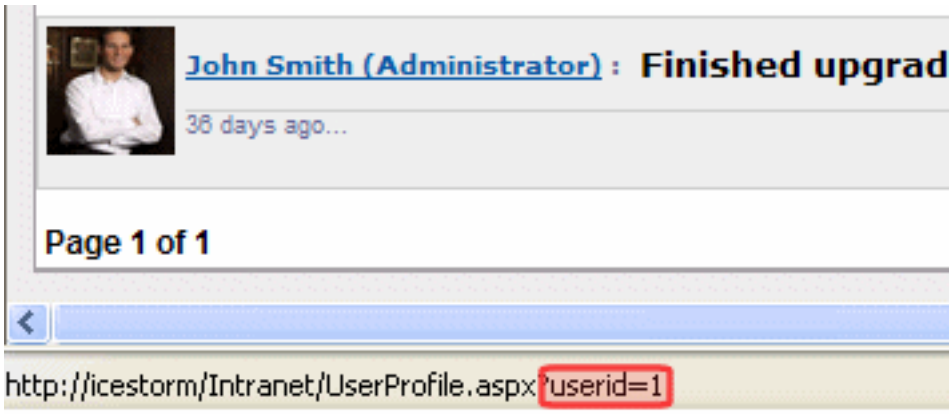
  This property, in conjunction with the **MaxNumber** property, lets site visitors view an unlimited number of items while controlling the amount of screen space. The **MaxNumber** property limits the number of items displayed. If you set this property to **True**, and the number of items exceeds **MaxNumber**, navigation aids appear below the last item, allowing the visitor to go to the next screen.
- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **MarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.See also: *Ektron Markup Language* on page 2633*Ektron Markup Language* on page 2633
- **MaxResults** (Integer)
  Enter the maximum number of items to appear in the initial display of this server control. To set no maximum, enter zero (0). To let site visitors view more than the maximum but limit the amount of space, enter the maximum number of results per page here. Then, set the **EnablePaging** property to **True**. If you do and more than the number of **MaxResults** are available, navigation aids help the site visitor view additional items.
- **Moderate** (Boolean)
  Set to `True` to force moderation of the Message Board. If set to `false`, moderation is controlled by the user or community group with which the Message Board is associated. See also: Moderating a Message Board.
- **ObjectType** (EkEnumeration.MessageBoardObjectType)
  The type of object to which the Message Board is assigned. Choices are:
  - **Content**. A content item
  - **User**. An individual
  - **CommunityGroup**. A community group
- **ProfileParamName** (String)
  The parameter name to pass in the QueryString to the TemplateUserProfile page, if you want it to be anything other than id. For example, you may prefer `userid`, because it is more descriptive, as shown in the following example.



- **ShowMaxCharacter** (Integer)

  The maximum number of characters to display in the comment. If the number of characters in a comment exceeds this, a **... more >>** link appears. Users click the link to see the full text.

- **SpamControlType** (EkEnumeration.MessageBoardSpamControlType)

  Assigns a spam filter to the Message Board. Ektron provides 3 spam filters, and the ability to define your own. Spam control is turned off by default. Filter choices are:

  - **SameUserMessageDay**. A user cannot post the same comment to a board more than once per day
  - **SameUserTimeDelay**. Prevents user from posting another comment for period of time, specified in the `SpamTimeSpan` property.
  - **SameMessageTimeDelay**. Prevents user from posting an identical comment for a period of time, specified in the `SpamTimeSpan` property.
  - **Custom**. Message Board uses your custom spam filter code. See Filtering Spam.

- **SpamTimeSpan** (Integer)

  The number of seconds for which the `SameUserTimeDelay` or `SameMessageTimeDelay` spam filters delay a user from posting a second time. See also: Filtering Spam.

- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.

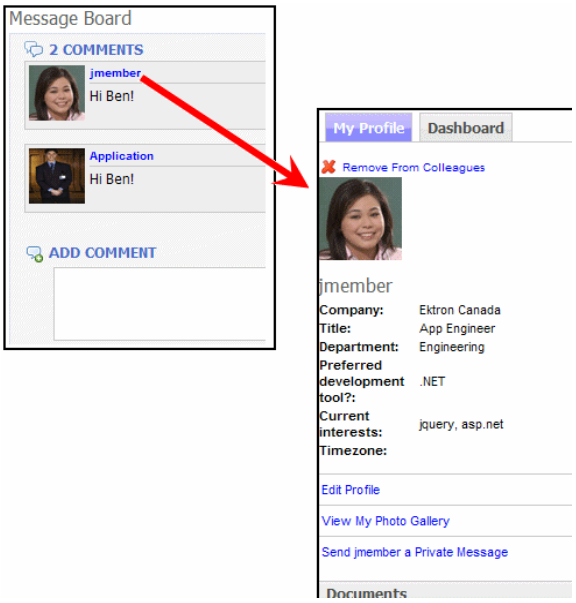
- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

- **TemplateUserProfile** (String)

  The URL path to a page that contains a UserProfile server control. The path can be relative or absolute. If you enter a path, a user can click any user's name or avatar from the Message Board server control and be forwarded to that profile page. See illustration below.



Note that user templates can be defined in the **Workarea > Settings > Community Management > Templates** screen. However, if you assign a template in this field, this setting takes precedence over the setting on the Workarea Template screen.

- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Messaging

8.50 and higher

The Messaging server control lets a site user send and receive messages. You can limit message recipients to a user's colleagues or expand it to all Ektron users. The Messages control can appear on a site or also in the Workarea. Typical messaging functions apply, such as compose (in the Workarea), reply (in the Workarea), forward (in the Workarea), print (in the Workarea), delete (in a sample site, in the Workarea).

Populating the To: field of a message

You can populate a message's **To:** field with user names by setting the `RecipientParamName` property to a QueryString parameter. This feature lets you create a link on a page, such as **Send a Message to this user** or **Send a message to all users in this group**. For example, you create a Web page link to a template containing the Messaging server control, pass a user's ID in the QueryString as UID,

and populate the `RecipientParamName` property with **UID**. As a result, the server control reads the user's ID and completes the **To:** field with the user's name.

Here is how the link might look: `messaging.aspx?g=pmessage&uid=1`

You can pass multiple user IDs. For example:
`messaging.aspx?g=pmessage&uid=1&uid=20&uid=12&uid=18`

IMPORTANT: You must include the `g=pmessage` parameter in the QueryString to open the Messaging server control to its editor. Otherwise, the server control opens to the Inbox.

The VB example below creates a dynamic hyperlink that populates a message's **To**: field with a logged in user's colleagues. For this example to work, add a Literal control to a Web form and name it **Lit1**. Notes are included as comment text.

```
Protected Sub Page_Load(ByVal sender As Object,
  ByVal e As System.EventArgs) Handles Me.Load

    Dim frnds As New Ektron.Cms.API.Community.Friends()
    Dim apicontent As New Ektron.Cms.ContentAPI
    Dim dirUsrData() As Ektron.Cms.DirectoryUserData

    If (apicontent.UserId > 0) Then
      dirUsrData = frnds.GetFriends(apicontent.UserId)
      If ((Not IsNothing(dirUsrData)) AndAlso (dirUsrData.Length > 0))
      Then
        Dim idx As Integer
        Dim friendsList As String = String.Empty
        For idx = 0 To dirUsrData.Length - 1
          'friendsList += "&uid=" + dirUsrData(idx).Id.ToString()
            friendsList = friendsList + "&uid="
            + dirUsrData(idx).Id.ToString()
        Next
          Lit1.Text = "<a href=""MsgSample.aspx?g=pmessage" + friendsList
            + """>Send a message to my friends</a>"
'
'Replace MsgSample.aspx with the Web form that contains
'the messaging server control.
'Set the RecipientParamName property to "uid" for this example.
'
        Else
            Lit1.Text = "The logged in user currently has no friends"
        End If
        Else
            Lit1.Text = "You must be logged in to run this demo"
        End If
        apicontent = Nothing
        frnds = Nothing
 End Sub
```

Inserting the Messaging server control onto a page

PREREQUISITE

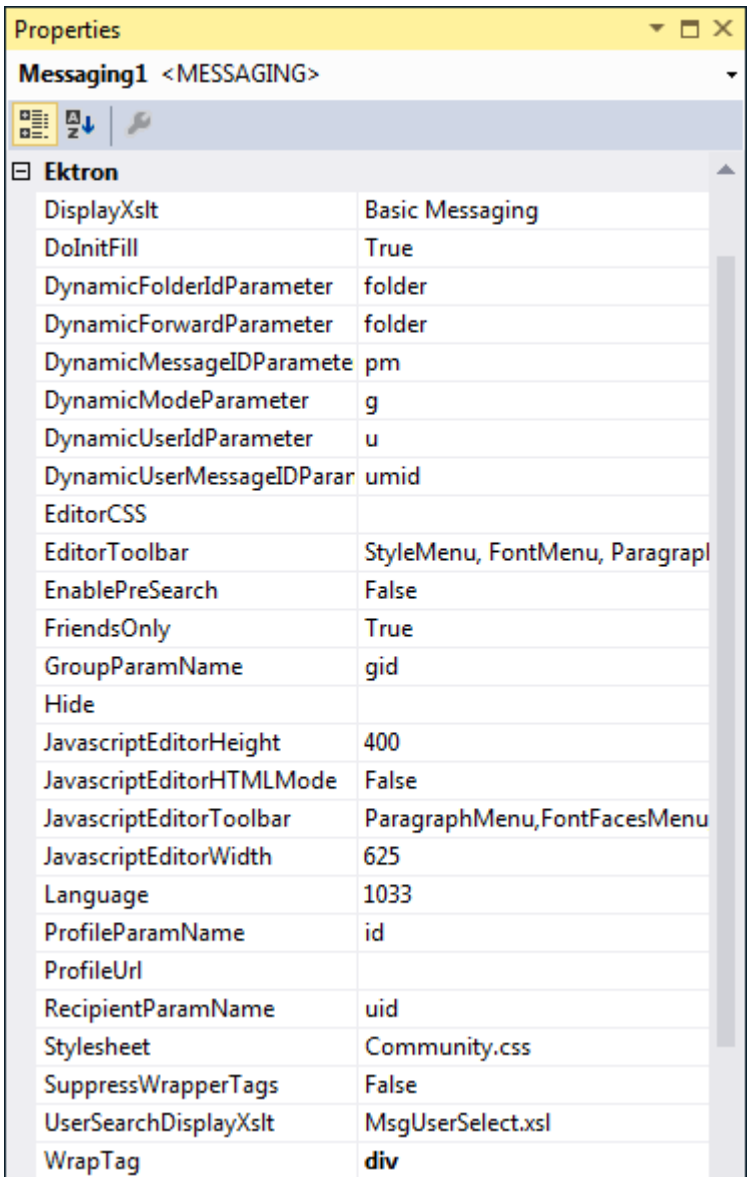
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Messaging** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Messaging ID="Messaging1" runat="server" />
```

4. Click on `Messaging` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



Messaging properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **EnablePreSearch** (Boolean)

  When set to True, all users are pre-loaded and displayed in the search results when browsing for users (an empty string search). If the `FriendsOnly` property is set to True, only colleagues are pre-loaded and displayed.

- **FriendsOnly** (Boolean)

  When set to True, users can only send messages to their colleagues.

  - **True** (default). Send messages to colleagues only
  - **False**. Send messages to all Ektron users

- **GroupParamName** (String)

  Enter the QueryString parameter used to pass the ID of a community group. The default is **gid**. Use this property to email all members of a group.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **JavascriptEditorHeight** (Integer)

  Set the height in pixels for the eWebEdit400 editor. The default is **300**.

- **JavascriptEditorWidth** (Integer)

  Set the width in pixels for the eWebEdit400 editor. The default is **360**.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **ProfileParamName** (String)

  The QueryString parameter used to pass the ID of a user to a profile page or location template when a user name is clicked in the Browse User screen. The default is ID. The parameter defined in this property is appended to the QueryString of the path defined in the `ProfileUrl` property. The default is ID.

  IMPORTANT: This parameter must match the parameter set in the UserProfile server control's `DynamicParameter` property on the page defined in the `ProfileUrl` property.

- **ProfileUrl** (String)

  The URL of the profile page template. This page opens in a new window when a user clicks a display name on the Browse Users screen. This provides additional profile information about the potential message recipient.

- **RecipientParamName** (String)

  Enter the QueryString parameter used to pass a user ID. For example, if the QueryString is `?uid=1`, enter `uid` in this property. To pass several users, enter a comma -separated list of user IDs. For example: `?uid=1,20,12,18`. This property is typically used when you want to populate a message's **To:** field. See also: *Populating the To: field of a message* on page 2276

- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

- **UserSearchDisplayXslt** (String)

  The XSLT used to display the user search control inside the Messaging server control. The default is `MsgUserSelect.xsl`.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

MicroMessaging

8.50 and higher

The MicroMessaging server control lets users post brief messages. It resembles micro-blogging services like Twitter.

Inserting the MicroMessaging server control onto a page

PREREQUISITE

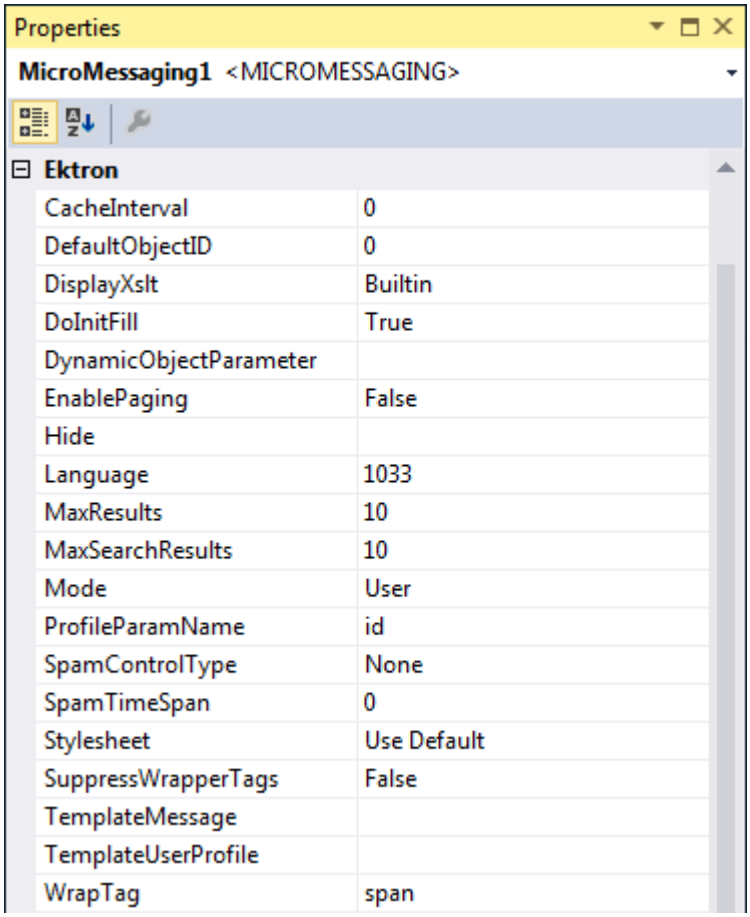
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **MicroMessaging** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:MicroMessaging ID="MicroMessaging1" runat="server" />
   ```
4. Click on `MicroMessaging` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as

you modify the property values.



MicroMessaging properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultObjectID** (Long)

  The default object ID for this control to use when there is no matching dynamic parameter value passed.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file,

create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicObjectParameter** (String)

  Gets or sets the QueryString parameter to read a object ID dynamically. To use the default object ID, leave blank. For example, if you are passing the ID value of a user, you might enter 'uid' for this property. So, if you passed `http://~yoursite~/UPpage.aspx?uid=21` on the QueryStrings to a page containing this control, you would see the list of Micro-messages for the user with an ID of 21.

- **EnablePaging** (Boolean)

  This property, in conjunction with the **MaxNumber** property, lets site visitors view an unlimited number of items while controlling the amount of screen space. The **MaxNumber** property limits the number of items displayed. If you set this property to **True**, and the number of items exceeds **MaxNumber**, navigation aids appear below the last item, allowing the visitor to go to the next screen.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **MaxResults** (Integer)

  Enter the maximum number of items to appear in the initial display of this server control. To set no maximum, enter zero (0). To let site visitors view more than the maximum but limit the amount of space, enter the maximum number of results per page here. Then, set the **EnablePaging** property to **True**. If you do and more than the number of **MaxResults** are available, navigation aids help the site visitor view additional items.

- **MaxSearchResults** (Integer)

  The maximum number of messages to return when performing a micro-message search.
  0 (zero) = unlimited.

- **Mode** (ActivityFeedType)

  Specify the type of micro-messages you want to display. Choices are:

  - **User**. Micro-messages for logged-in user with search option
  - **Colleagues**. Micro-messages for logged-in user and colleagues of a specified user with search option
  - **TimeLine**. Micro-messages from all site users whose Private Profile setting is set to Public; includes search option
  - **Message**. A single message
- **ProfileParamName** (String)

  The parameter name to pass in the QueryString to the TemplateUserProfile page, if you want it to be anything other than id. For example, you may prefer `messageid`, because it is more descriptive.
- **SpamControlType**

  Assigns a spam filter to the micro-messages. The control supplies 3 predefined spam filters, and the ability to define your own. Spam control is turned off by default. Filter choices are:

  - **SameUserMessageDay**. Prevents a user from posting the same message to the board more than once per day
  - **SameUserTimeDelay**. Prevents a user from posting another message for specified period of time. The amount of time is specified, in seconds, in the `SpamTimeSpan` property.
  - **SameMessageTimeDelay**. Prevents a user from posting an identical message for a specified period of time. The amount of time is specified, in seconds, in the `SpamTimeSpan` property.
  - **Custom**. Control uses custom spam filter code. See Filtering Spam.
- **SpamTimeSpan** (Integer)

  Sets the amount of time, in seconds, for which the `SameUserTimeDelay` or `SameMessageTimeDelay` spam filter delays a user from posting. These values can be selected in the `SpamControlType` property. See also: .
- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.
- **SuppressWrapperTags** (Boolean)

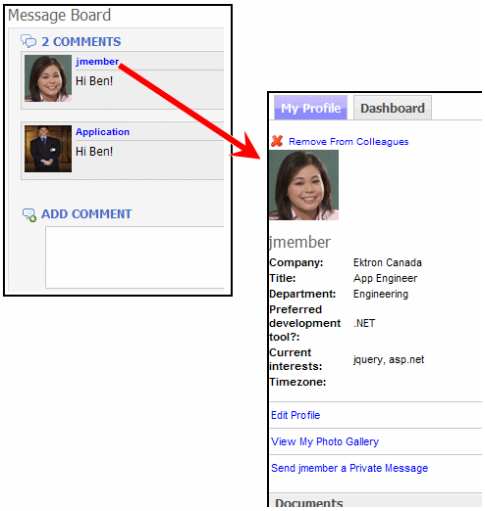
  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TemplateMessage** (String)

  The URL path to a page that contains another Micro-messaging server control. When this property contains such a path, and the destination page has a Micro-messaging control whose `Mode` property is set to `Message`, a user can click a micro-message's time span on the first page to open a second page that contains just that message.See also: .

- **TemplateUserProfile** (String)

  The URL path to the page that contains the UserProfile server control. The path can be relative or absolute. If you enter a path, a user can click any user's name or avatar from the Micro-message server control and be forwarded to that profile page.



  Note that user templates can be defined in the **Workarea > Settings > Community Management > Templates** screen. However, if you assign a template in this field, this setting takes precedence over the setting on the Workarea Template screen.

- **WrapTag** (String)

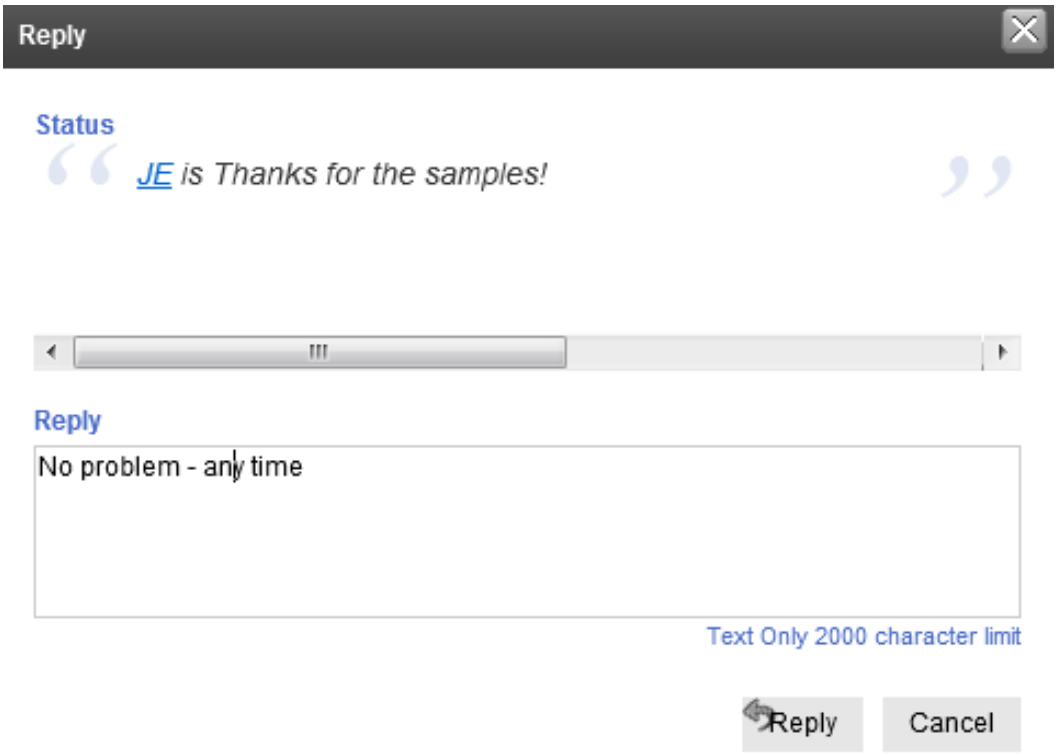
  Lets a developer specify a server control's tag.

  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

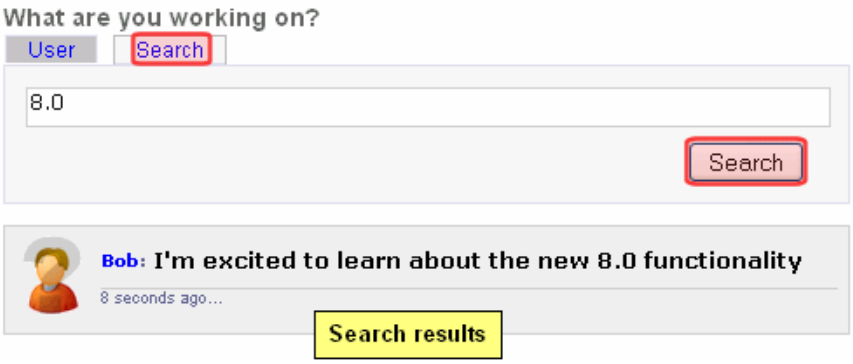
Using the Micro-messaging server control

The Micro-messaging server control allows users to post brief messages. It resembles micro-blogging services like Twitter.

A user can reply to another user's micro-message.



The control also lets users search micro-messages. See also: *Searching for micro-messages* on page 2293



In addition, micro-messages are a type of notification.

This feature builds tighter-knit communities by allowing members to share information and feel they have a personal stake in that information.

NOTE: If you are in a load-balanced environment, the same reply may appear several times. If this problem occurs, stop the Ektron notification service on all but one of the servers in the cluster.

This section also contains the following topics.

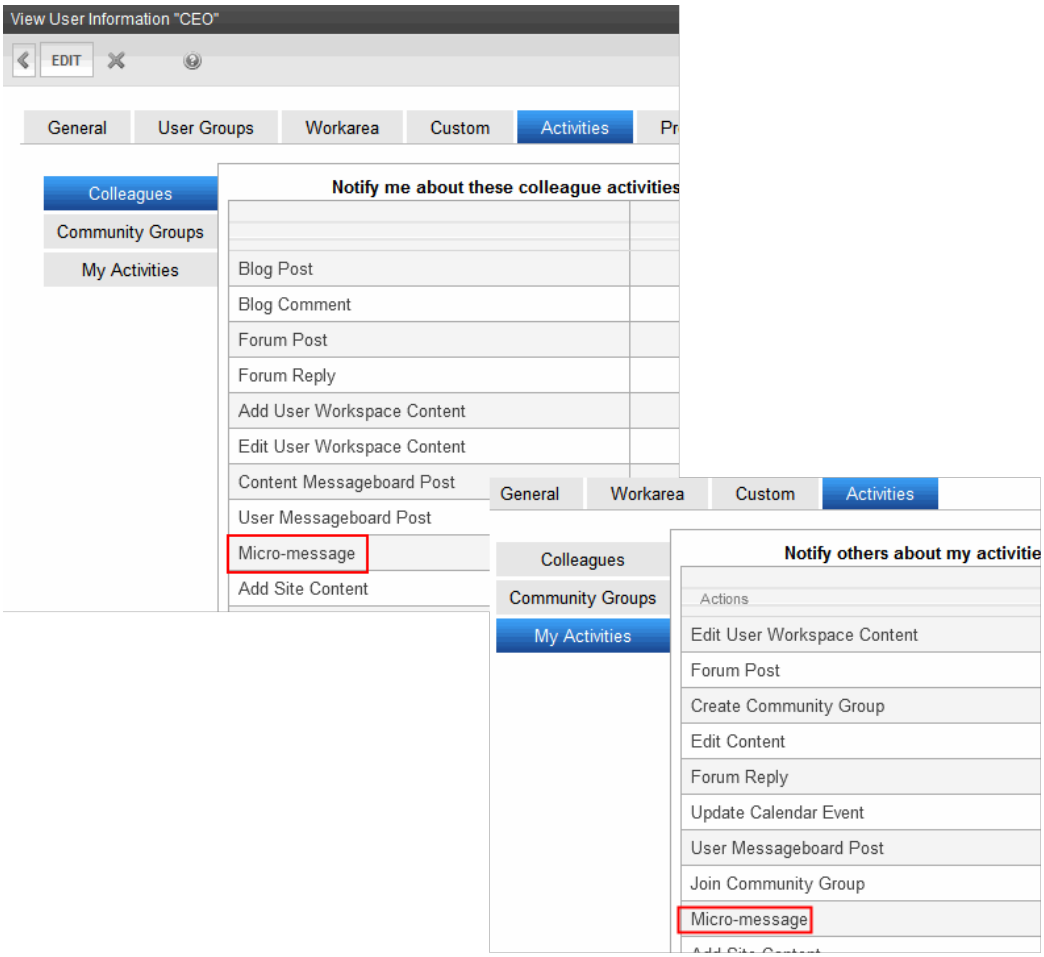
NOTE: A user's **Private Profile** setting has 3 possible values: **Public**, **Private** and **Colleague**. When discussing this control and these settings, **Private** and **Colleague** both act as **Colleague**.

Micro-messages are an activity type

A micro-message is type of Ektron activity, like adding content or joining a Community Group. As such, a new micro-message can generate a notification. See also: Sending Notifications to a Community.

Like other notifications, micro-messages can appear on an Activity Stream control.

Users can determine if they want to be notified about colleagues' micro-messages on the **User Profile > Activities > Colleagues** tab, and if want to notify others on the **My Activities** tab.



Micro-messaging display modes

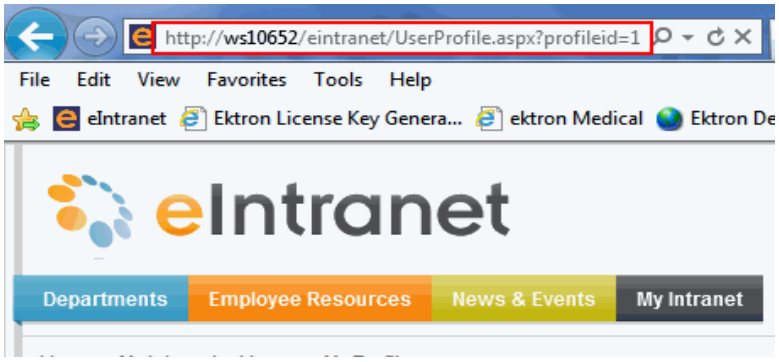
This Micro-messaging server control has 4 modes, which let you control the messages being displayed.

Associating the control with a user

Like other Ektron server controls, there are 2 ways to specify the user associated with the control.

- the server control's `DefaultObjectID` property. Do this to display a particular user's micro-messages on a page.
- the user ID is retrieved from the page URL, which typically contains the logged-in user ID (see example below). Do this to display micro-messages of the

logged-in user.

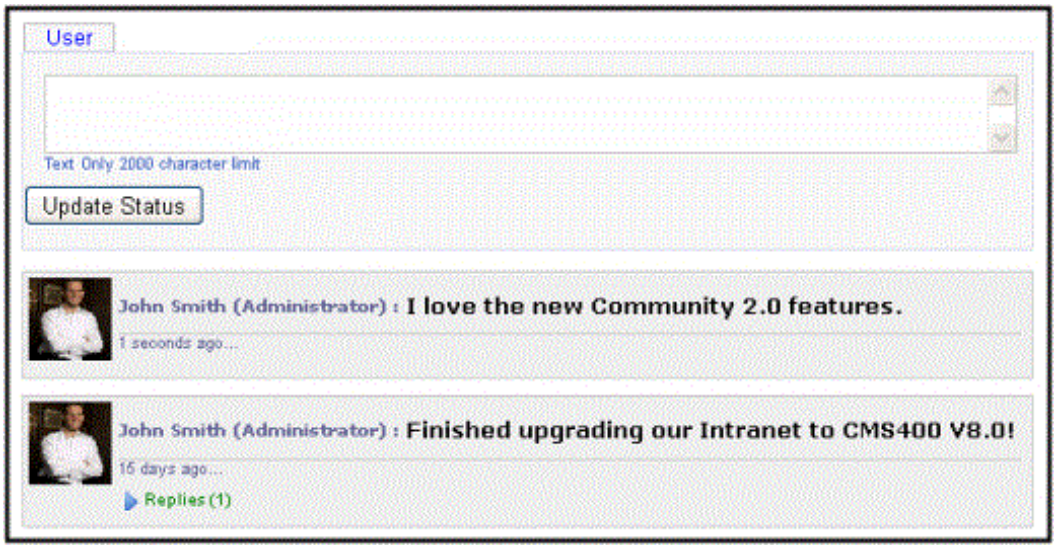


User mode

Use this mode to show micro-messages for a specified user, such as on a User Profile page. For example, when you view your own page in User Mode, you can

- see your messages
- submit a new micro-message. For example, you update your status.
- search micro-messages. See also: *Searching for micro-messages* on page 2293.

The following example of the Micro-messaging server control with the Mode property set to `User`.



In User Mode, you can view the control that is associated with another user. For example, when you view another's profile, which contains the Micro-messaging control in User Mode, you see only micro-messages of the user who owns the profile. You cannot submit a micro-message from your colleague's profile page, nor can you search micro-messages. For example, you and Steve are colleagues. When you view Steve's profile page:

- you can see all of Steve's messages
- you can see messages from colleagues they have in common
- you *cannot* update Steve's status
- you *cannot* search messages

Colleagues mode

Colleagues Mode displays micro-messages from the user associated with the control *and the user's colleagues*. Use this mode to show micro-messages for these users, such as on a user profile page. When you are logged in and visit a page containing a Micro-messaging control in colleagues mode, you can:

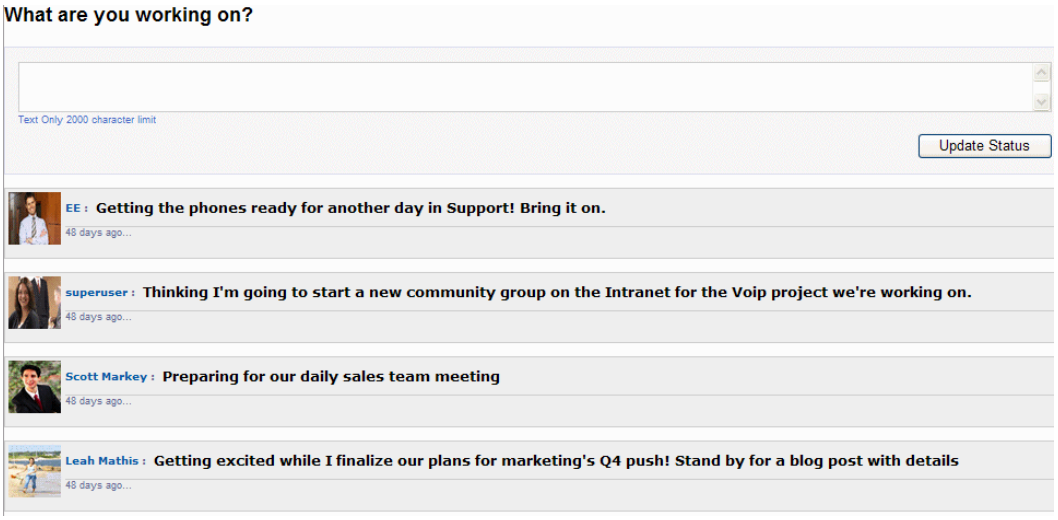
- see the colleague's messages
- see the colleagues' messages from other colleagues
- update the colleague's profile
- search micro-messages. See also: *Searching for micro-messages* on page 2293

You also can view the control in Colleagues Mode when it is associated with another user. For example, when you view another user's profile, which contains the Micro-messaging control in Colleagues Mode, you see only micro-messages from the user who owns the profile and the colleagues that you have in common with the user. You cannot submit a micro-message from the colleague's profile page, nor can you search micro-messages. For example, you and Steve are colleagues. When you view Steve's profile page:

- you can see Steve's messages
- you can see messages from colleagues you have in common
- you *cannot* update Steve's status
- you *cannot* search messages

TimeLine mode

This mode displays micro-messages for a site's users whose **Private Profile** setting is set to **Public**. This mode displays a chronological stream of micro-messages from all such users, with the most recent at top.

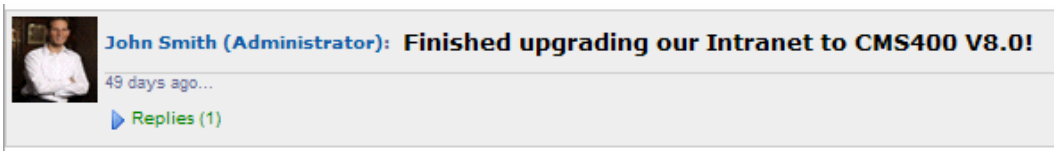


Whenever a user whose profile is set to Public submits a micro-message, it's added to the time line.

From the time line page, logged in users can reply to and search micro-messages. See also: *Replying to a micro-message* on page 2295, *Searching for micro-messages* on the facing page

Message mode

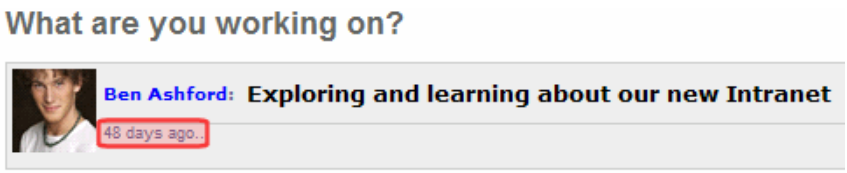
This mode displays a single micro-message on a page.



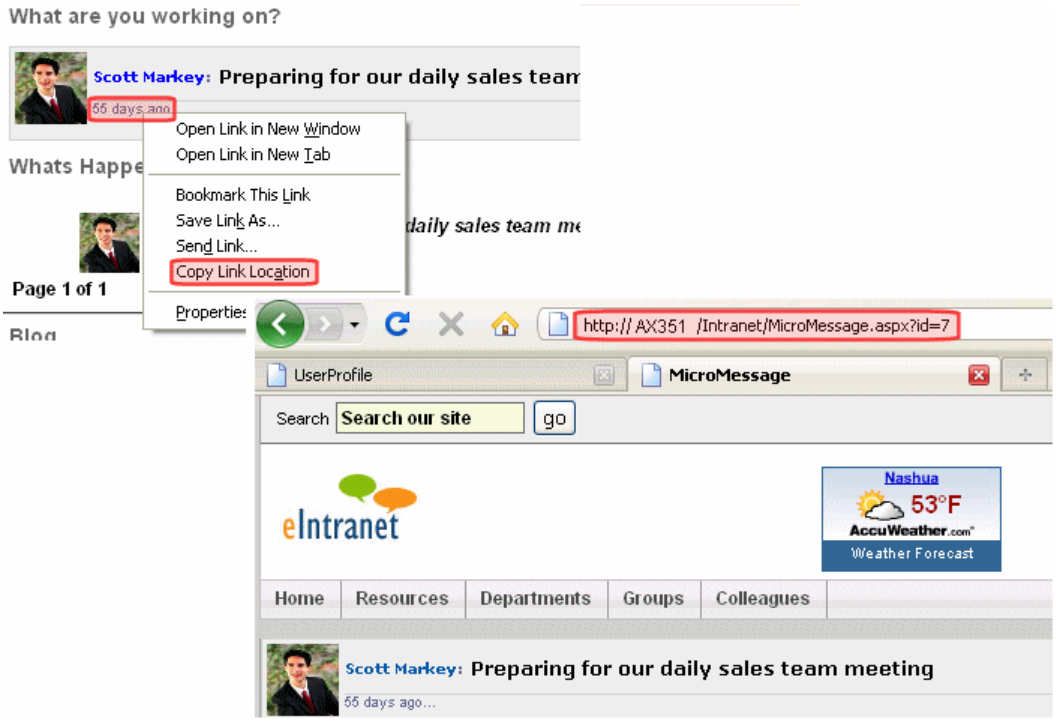
For example, John sends his boss a link to a micro-message that praises his work. When John's boss clicks the link, a page that contains this control appears. See also: *Making a hyperlink to a single-page version of a micro-message* below

Making a hyperlink to a single-page version of a micro-message

Each micro-message includes a *time lapse*, that is, the length of time since a message was submitted. See example below.



Each micro-message's time lapse can become a hyperlink that contains the ID of that message and a destination window. Users can click the link to view that message in the new window.



The destination window must

- be identified in the properties of the Micro-messaging server control that displays the original message
- contain a Micro-messaging server control in Message Mode

Users can also copy the link and send it to another user, who can then view the message. To copy a time lapse's link information, hover over the link, right click, and select **Copy Shortcut** (if using Internet Explorer) or **Copy Link Location** (if using Firefox).

NOTE: If someone sends a micro-message link to another user who is ineligible to see the message, it does not appear. To see the message, either the user who submitted it has the profile set to **Public,** or the link recipient must be a colleague of the message's creator.

The following example shows how to set up this capability.

- `Page1.aspx` contains a Micro-messaging server control that displays a user's micro-messages. The messages' time lapses are hyperlinks.
- `Page2.aspx` has a Micro-messaging server control in Message Mode. It displays the individual message whose time lapse was clicked on the first page.

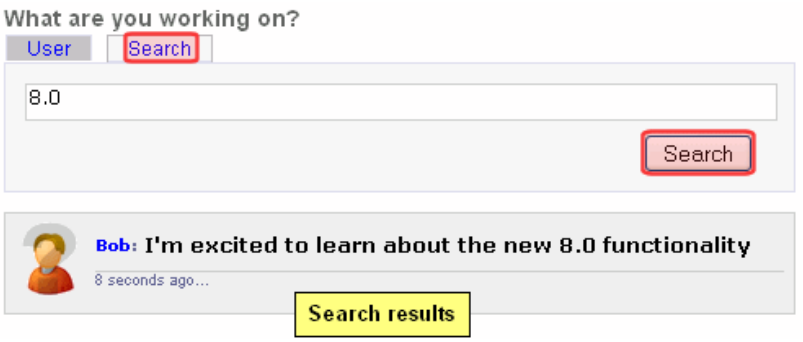
NOTE: For a full description of the micro-message properties, see *MicroMessaging properties* on page 2282.

1. On `Page1.aspx`, add a Micro-messaging server control.
2. Set the `DynamicObjectParameter` to `id`.
3. Set the `TemplateMessage` to the path to `Page2.aspx`.
4. Set the `Mode` property to anything other than `Message`.
5. Save `Page1.aspx`.
6. On `Page2.aspx`, add a Micro-messaging server control.
7. Set the `DynamicObjectParameter` to `id`.
8. Set the `Mode` property to `Message`.
9. Save `Page2.aspx`.

Now, when you log in to your site and navigate to `Page1.aspx`, you see a list of your micro-messages. If you click the time lapse on any message, `Page2.aspx` loads, showing the message you clicked.

Searching for micro-messages

The Micro-messaging server control includes search functionality that prompts a user to enter terms and returns micro-messages that match them. The search looks through micro-messages only. It does *not* search content, products, users, or groups.



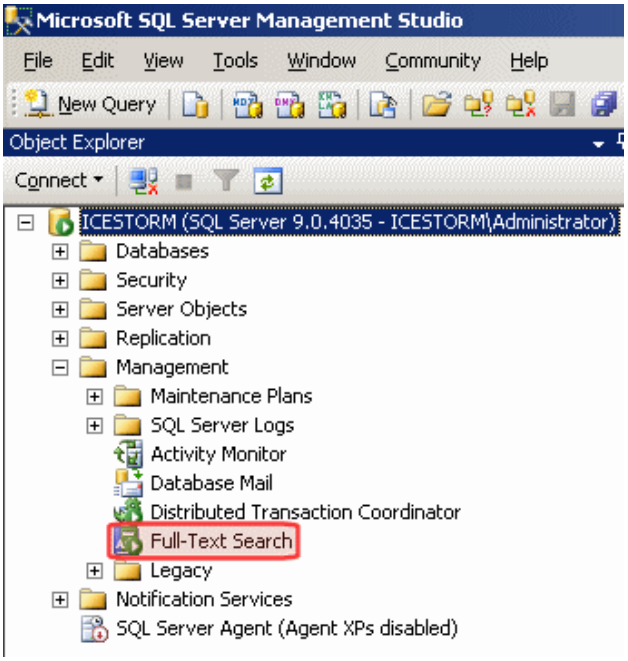
When searching, a user can only find micro-messages submitted by

- one's self
- one's colleagues
- users whose profile is set to public

The following topics explain how to enable and use the micro-message search.

Enabling the micro-message search

The micro-message search requires your server to have the Full Text Search component of Microsoft SQL Server.



You can install the Full Text Search component during the installation of Microsoft SQL Server.

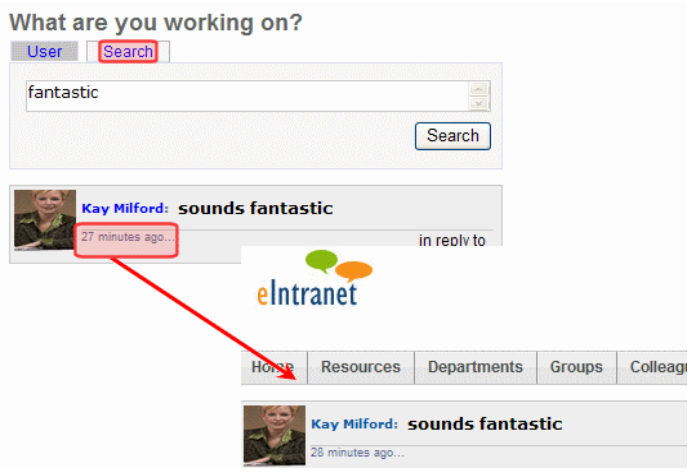
If the Full Text Search component is not installed, and you want to use the Micro-messaging search functionality, insert the SQL Server installation disc into your computer and look for **Full Text Search**.

After the Full Text Search is installed, run the FullTextIndex.sql script on your database. This script is located in:

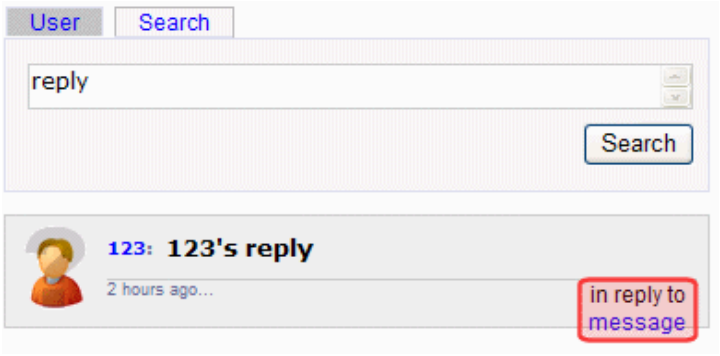
```
Program Files\Ektron\CMS400vxversionnumber\
  Utilities\SiteSetup\Database\FullTextIndex.sql
```

Running a micro-message search

When micro-message search is enabled, a **Search** tab appears on the control. To use the search, click the tab and enter search terms.



When results appear, you can click any message's time span to proceed to the message itself. If a reply appears among search results, you can click the link indicated below to view the original message.



Entering multiple search terms

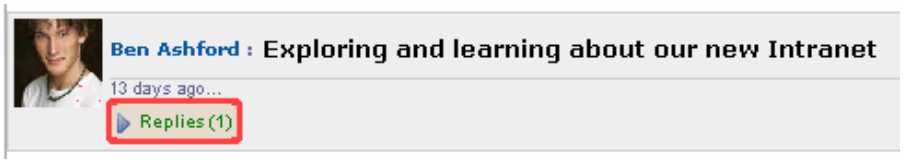
You can enter more than one search term. If you do, the terms have an "and" relationship. For example, if you enter **Community** and **Group**, only messages containing *both* Community *and* Group are found.

Replying to a micro-message

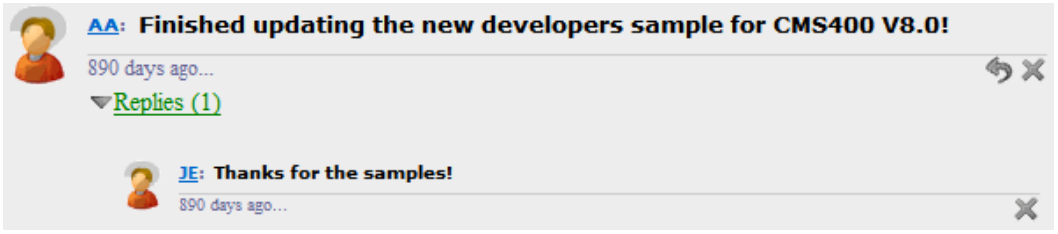
The micro-message control lets users reply to micro-messages and nests the reply with the original message. Anyone who can see a message can reply to it, and see all other replies.

To add a reply, hover over a message and click **Reply** ( ). When you do, a dialog box prompts you to submit a reply of up to 2000 characters. The box also displays the original message.

Replies do not appear when a micro-message is viewed. Instead, a **Replies** link appears below the comment. The number of replies appears next to the link.



Click **Replies** to view them. The newest replies are at the top.



To remove a reply, click **Delete** ( ), which appears when you hover over the reply. Only the user who submitted the reply and the message board owner can delete replies.

NOTE: If you are in a load-balanced environment, the same reply may appear several times. If this problem occurs, stop the Ektron notification service on all but one of the servers in the cluster.

Filtering micro-message spam

You can filter spam from micro-messages. Ektron defines spam filtering as preventing users from posting a micro-message or replying using:

- the same message more than once a day
- any second message within a specified time period after posting a first
- the same message within a specified time period

NOTE: If a user enters a micro-message and a reply that have the same text, each can be sent once without being stopped by the spam filter. However, if either is sent a second time within the specified time period, the spam filter blocks it.

The Micro-messaging server control also lets you define custom spam filters.

To define a spam filter, use the `SpamControlType` property on the MessageBoard server control. The table below provides possible values and examples of how to set that property.

Ektron lets you filter spam from your message boards. Ektron defines *spam filtering* as setting the following limits in MessageBoard server control's SpamControlType property:

- **SameUserSameMessageSameDay**. Prevent a user from posting the same comment more than once a day.
- **SameUserTimeDelay**. Set the SpamTimeSpan property to 1800 to prevent a user from posting any other comment for 30 minutes.
- **SameMessageTimeDelay**. Set the SpamTimeSpan property to 7200 to prevent a user from posting identical comments for 2 hours.
- **Custom** (and add the code to the code-behind page of the template containing the MessageBoard server control). Use your own spam filter code.

Creating a custom spam filter

You can create your own spam filter in the code-behind of the page that contains the Micro-messaging control. To do this, set the `SpamControlType` property to `Custom`, call the `CustomSpamMethod` in the page load event, and point it to your custom spam filter method.

The following Visual Basic example blocks the text "Hello World."

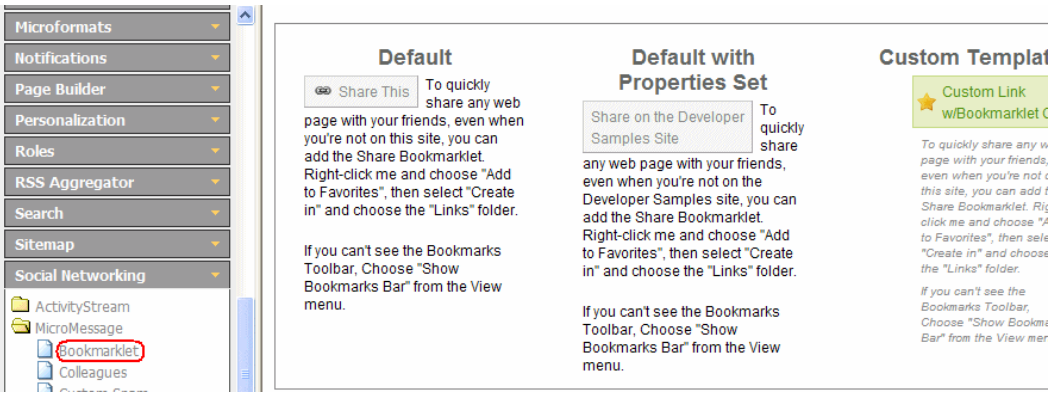
```
                                        Partial Class Default
Inherits System.Web.UI.Page
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Load
Me.MicroMessaging1.CustomSpamMethod(AddressOf SpamHandler)
End Sub
Private Function SpamHandler(ByVal data As   Ektron.Cms.MicroMessageData) As Boolean
If data.MessageText = "Hello World" Then
Return True
End If
```

```
End Function
End Class
```

MicroMessagingBookmarklet

8.60 and higher

The MicroMessagingBookmarklet server control detects the user's browser and modifies the text accordingly. For more information, see Using Ektron's MicroMessagingBookmarklet.



Inserting the MicroMessagingBookmarklet server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

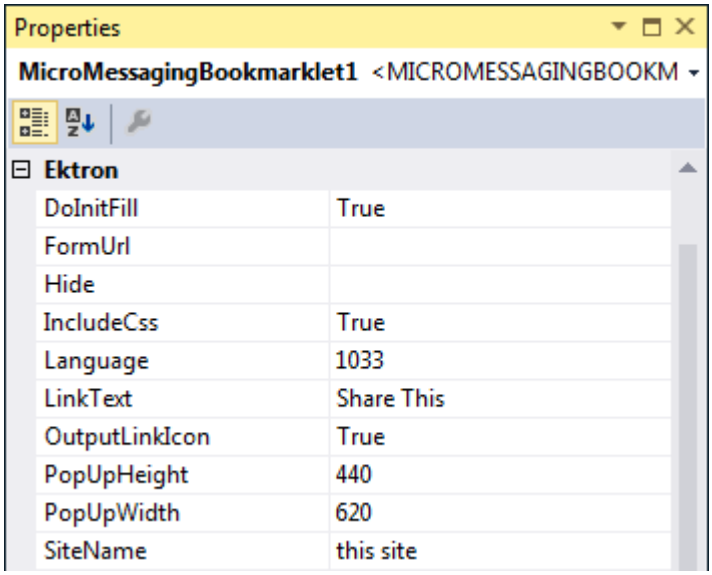
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **MicroMessagingBookmarklet** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:MicroMessagingBookmarklet ID="MicroMessagingBookmarklet1" runat="server" />
```

4. Click on `MicroMessagingBookmarklet` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is

updated as you modify the property values.



MicroMessagingBookmarklet properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **FormUrl** (String)

  The path to the page that appears when a user clicks the MicroMessagingBookmarklet link. By default, when a user clicks a link, he is redirected to the `siteroot/Workarea/share.aspx` page.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

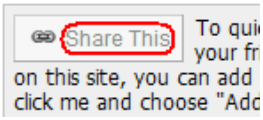
- **IncludeCSS** (Boolean)

  Lets you include or exclude the control's default .css file, which provides the "look and feel" of the control on a Web page.

- **Language** (Integer)

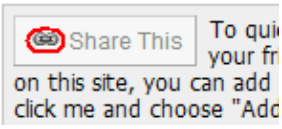
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LinkText** (String)
  The text that appears on the MicroMessagingBookmarklet link.



  To modify the look and feel of this area, edit the ektron.micromessaging.bookmarklet.css file in the *siteroot*/Workarea/csslib folder.

- **OutputLinkIcon** (Boolean)
  - **False**. Suppress the icon
  - **True**. Show the MicroMessagingBookmarklet link icon on the button

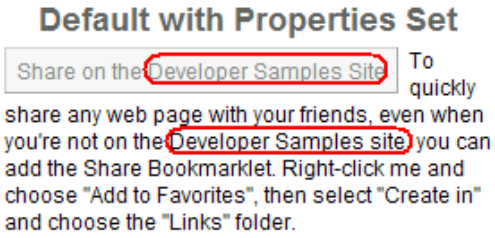


  To modify the look and feel of this icon, edit the ektron.micromessaging.bookmarklet.css file in the *siteroot*/Workarea/csslib folder.

- **PopupHeight** (Integer)
  The height of the MicroMessagingBookmarklet link window in pixels.

- **PopupWidth** (Integer)
  The width of the MicroMessagingBookmarklet link window in pixels.

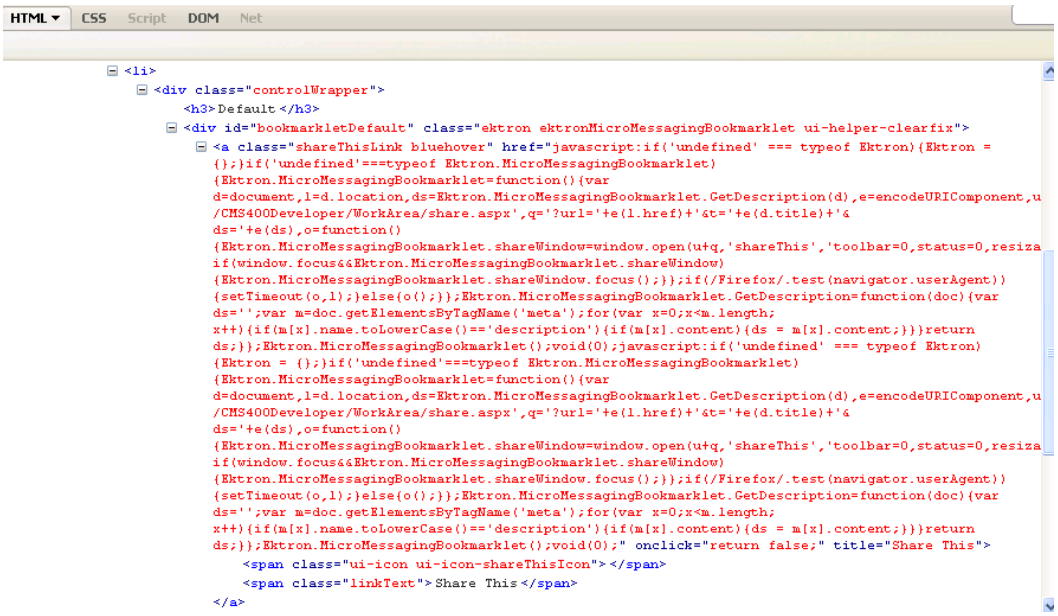
- **SiteName** (String)
  The name of the website using this server control. The SiteName value is embedded in the instructions for adding the MicroMessagingBookmarklet to the toolbar. See example below.



Customizing the MicroMessagingBookmarklet server control

The MicroMessagingBookmarklet server control is template-based which means you can change the control's appearance without an extensive XSLT or EkML file.

The JavaScript used to render the MicroMessagingBookmarklet control is loaded into the `Href` property of the control's anchor tag.



The following code shows the first MicroMessagingBookmarklet server control in `bookmarklet.aspx`.

```
<li>
<div class="controlWrapper">
<h3>Default</h3>
<CMS:micromessagingbookmarklet ID="bookmarkletDefault" runat="server" />
</div>
</li>
```

Because MicroMessagingBookmarklet is a templated server control, you can enter any valid markup, and it will be rendered exactly as you entered it. As examples, you can enter the following items within the `<div>` tags that surround the control.

- another Ektron server control
- any third-party server control
- HTML syntax
- databinding syntax
- style classes
- style tags, such as `<strong>`

PhotoGallery

8.50 and higher

IMPORTANT: Deprecated. The PhotoGallery server control is deprecated. If you are already using a deprecated item, you can continue to do so, but Ektron recommends using current versions of functionality.

The PhotoGallery server control helps users and community groups track and manage images.

Inserting the PhotoGallery server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

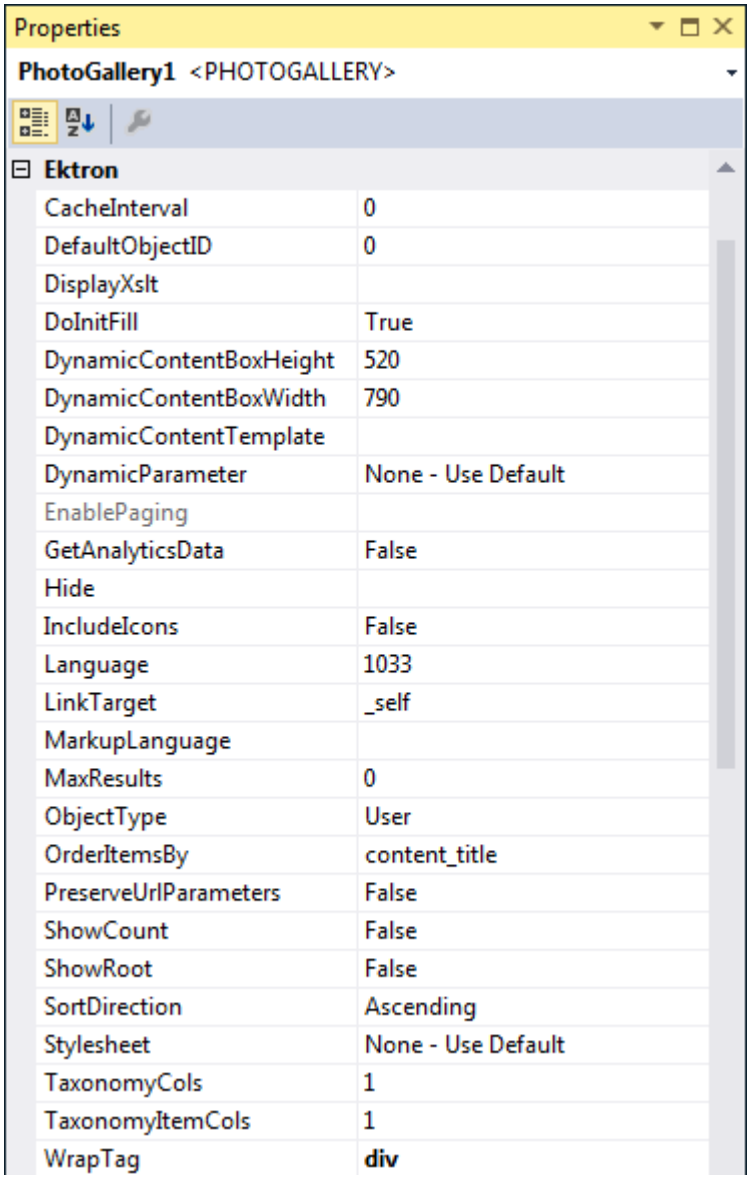
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **PhotoGallery** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:PhotoGallery ID="PhotoGallery1" runat="server" />
   ```

4. Click on `PhotoGallery` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



PhotoGallery properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultObjectID** (Long)

  The default object ID for this control to use when there is no matching dynamic parameter value passed.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicContentBoxHeight** (Integer)

  The height of the dynamic content box in pixels.

- **DynamicContentBoxWidth** (Integer)

  The Width of the dynamic content box in pixels.

- **DynamicContentTemplate** (String)

  The template to use when displaying dynamic content. Leave blank to use the dynamic box.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a object ID dynamically. To use the default object ID, leave blank.

- **EnablePaging** (Boolean)

  This property, in conjunction with the **MaxNumber** property, lets site visitors view an unlimited number of items while controlling the amount of screen space. The **MaxNumber** property limits the number of items displayed. If you set this property to **True**, and the number of items exceeds **MaxNumber**, navigation aids appear below the last item, allowing the visitor to go to the next screen.

- **GetAnalyticsData** (Boolean)

  Set this property to **True** if you want the following information for each content in the list. Returns **Content View Count**, **Content Rating**, **Content Rating Average**. Create your own XSLT styles to display this data.

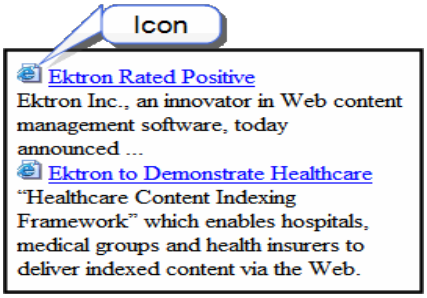
  IMPORTANT: This property provides reliable data only when the Business Analytics Feature is on. See Running Ektron Business Analytics.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **IncludeIcons** (Boolean)
  Choose whether to display icons next to the navigation list's links.



  - **True**. Show icons
  - **False**. Hide icons
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **LinkTarget** (String)
  Determines the type of window that appears when you click a link in the server control.
  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **MarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.

  The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.

  To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.

  See also: *Ektron Markup Language* on page 2633
- **MaxResults** (Integer)
  Enter the maximum number of items to appear in the initial display of this server control. To set no maximum, enter zero (0). To let site visitors view more than the maximum but limit the amount of space, enter the maximum number of results per page here. Then, set the **EnablePaging** property to **True**. If you do and more than the number of **MaxResults** are available, navigation aids help the site visitor view additional items.
- **ObjectType** (String)
  The type of object to which this control is assigned. Choices are:

  - User
  - Group
- **OrderItemsBy**

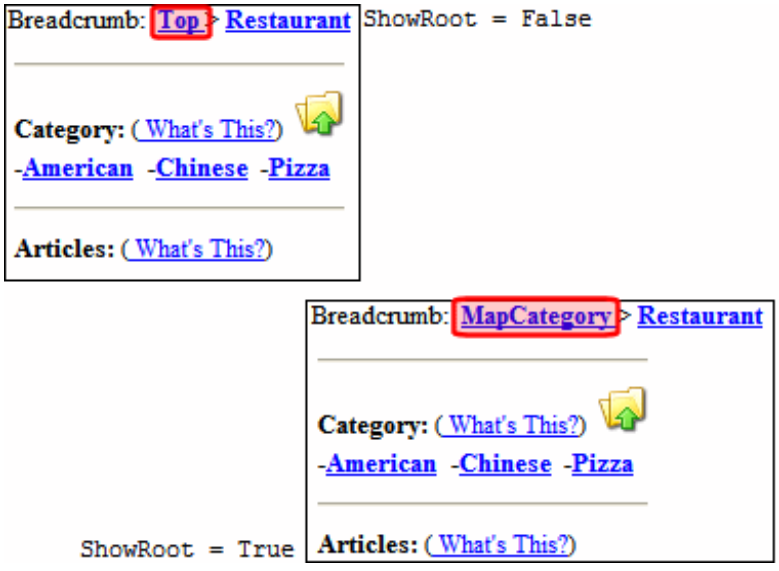
  Specify the sort order of results. Choices are:

  - **taxonomy_item_display_order**. The order of taxonomy items as set in the Workarea. For additional information, see Reordering Content Assigned to a Taxonomy Category.
  - **content_title**. Alphabetical order by title
  - **date_created**. Chronological order by date created
  - **last_edit_date**. Chronological order by date edited

  To specify the direction of the items, use the `SortDirection` property.
- **ShowCount** (Boolean)

  Indicates if an integer representing the number of taxonomy items in the category appears next to each category when displayed on the website. The default is False.

  - **True**. Show taxonomy items number
  - **False**. Do not show taxonomy items number
- **ShowRoot** (Boolean)
  - **False. Top** represents the first node of the taxonomy path.
  - **True**. The *name* of the taxonomy path's first node appears instead of **Top**.



- **SortDirection** (String)

  Select the direction of the `itemSortOrder` property. Choose Ascending or Descending.
- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder

outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.

- **TaxonomyCols** (Integer)
  Enter the number of columns in which this taxonomy/category appear on the page.
- **TaxonomyItemCols** (Integer)
  Enter the number of columns in which the taxonomy items appear on the page.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Using the PhotoGallery server control

IMPORTANT: The PhotoGallery server control is deprecated. If you are already using a deprecated item, you can continue to do so, but Ektron recommends using current versions of functionality.

The PhotoGallery server control helps users and community groups track and manage images. They can organize the images in any number of taxonomy categories. For example, a user might have the following organization.

- Science
  - Biology
    - Animals
      - Mammals
        - Lions
        - Bears

After you upload an image to the a Photo Gallery, site visitors can navigate through its taxonomy and view the image.

When you associate this control with a community group, you can allow group members to add, remove and control the sharing of folders in a workspace. Because this feature is implemented on a group-by-group basis, it is controlled in the Workarea's **Edit Community Group** screen. See also: Allowing Community Group Members to Work with Folders.

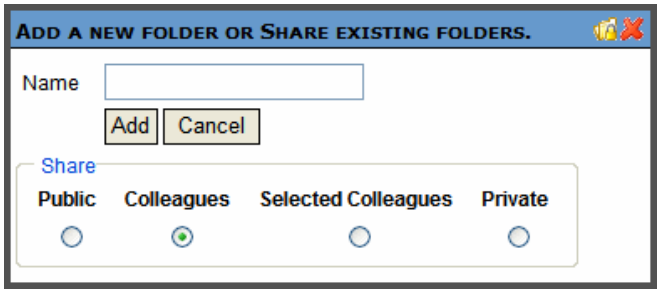
Categories

This section contains the following topics.

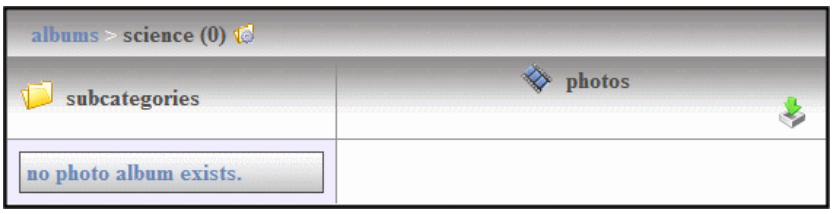
Adding a category

You can add categories to your photo gallery to sort your photos. You also can use categories to identify the types of users who can view the photos. For example, one category of photos may be private, while another may be shared with all of your colleagues.

1. On the website, navigate to a **User's Profile Page > Photo Gallery**.
2. Click **Manage Folder** ( ). The Add box appears.



3. Enter a **Name** for the folder.
4. In the **Share** area, select with whom you want to share your photos.
   - **Public**. Anyone who can access your profile
   - **Colleagues**. Only colleagues (either you accepted their invitation, or they accepted yours)
   - **Selected Colleagues**. Colleagues that you designated as *selected*. See also: *Designating a selected colleague* on page 2266.
   - **Private**. Only you
5. Click **Add**. When the page refreshes, it opens to the newly added folder.



Renaming a category

1. On the website, navigate to a **User's Profile Page > Photo Gallery**.
2. Click **Edit** ( ), located to the left of the category's title. The Edit box appears.

3. Change the name of the category.
4. Click **Save**.

Deleting a category

WARNING! Deleting a category permanently deletes all photos, as well as its subcategories.

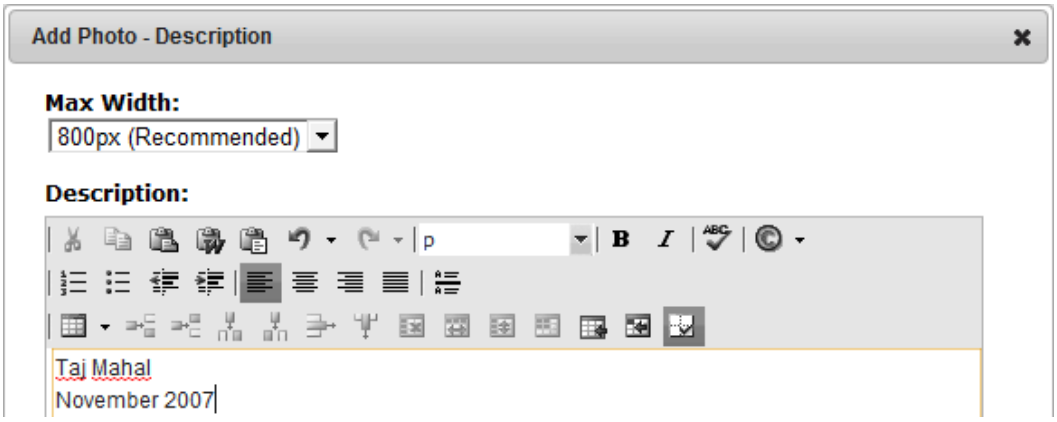
NOTE: You can not delete the top-level folder, Albums.

1. On the website, navigate to a **User's Profile Page > Photo Gallery**.
2. Click **Edit** (), located to the left of the category's title. The Edit box appears.
3. Click **Delete**. A dialog box appears asking you to confirm.
4. Click **OK**.

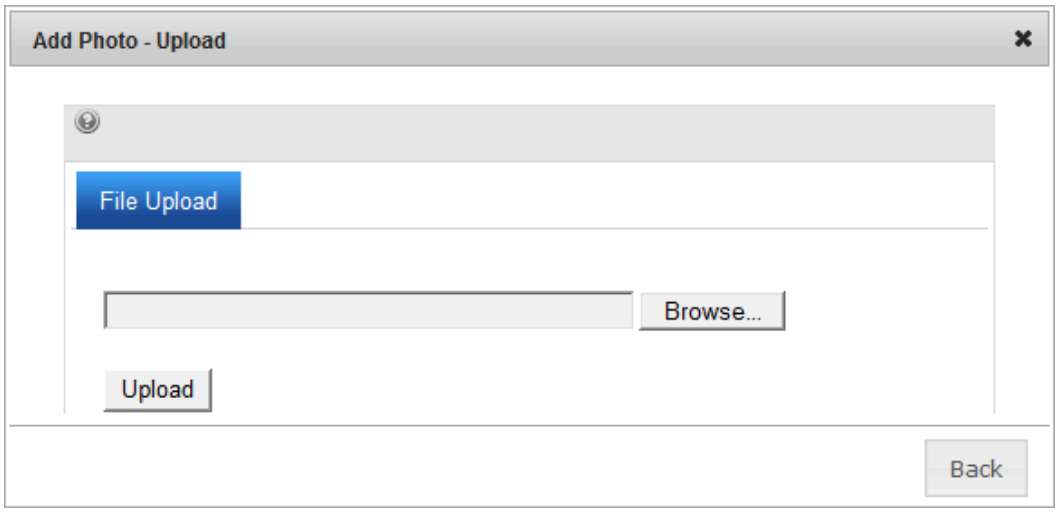
Photos

Adding Photos to a photo gallery

1. On the website, navigate to a **User's Profile Page > Photo Gallery**.
2. Select a category where the photo will be added. If you want to create a new category, see *Adding a category* on the previous page
3. Click **Add Photo**. The Add Photo box appears
4. In the Photo Data area, enter a Description and Maximum Width.



5. Click **Next**. The Add Photo dialog appears. Its appearance varies according to your browser, as explained in Storing External Files as Library Items vs. Assets.



6. User the **Browse** or **Choose File** button to navigate to the photo in your file system. Or, you may be able to click the **Drag Drop** tab and drop a photo in the Add box. A status box shows the files being uploaded. The page refreshes, and the photo appears in the gallery.

Changing a photo title or description

1. On the website, navigate to a **User's Profile Page > Photo Gallery**.
2. Select the category which contains the photo you want to change.
3. Click the triangle to the right of the photo's title and choose **Edit Properties**.
4. Edit the photo's Title, Description, or both.
5. Click **Save**.

Moving and copying photos

1. On the website, navigate to a **User's Profile Page > Photo Gallery**.
2. Select the category from which to move or copy the photo.
3. Click the triangle to the right of the photo's title and choose **Copy**.
4. Select the category to which the photo will be moved and click **Manage**.
5. Click **Move** or **Copy**. A dialog box asks you to confirm the action.
6. Click **OK**.

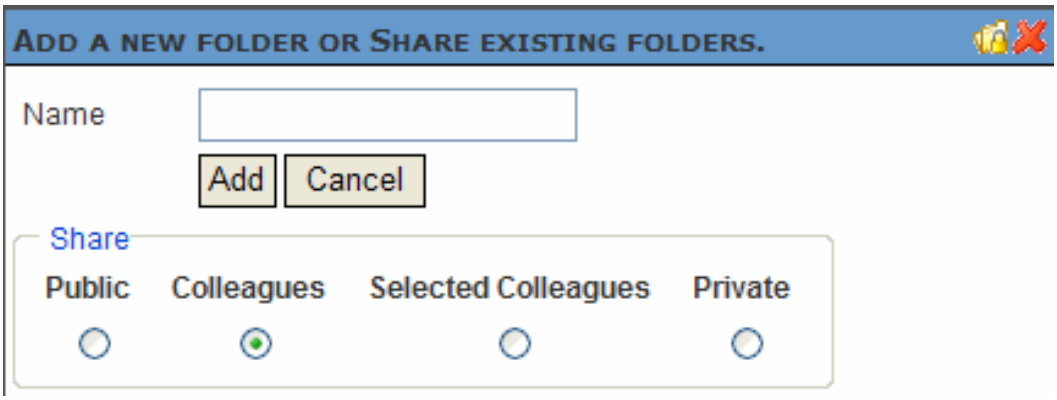
Deleting a photo

1. On the website, navigate to a **User's Profile Page > Photo Gallery**.
2. Select the category which contains the photo you want to delete.
3. Click the triangle to the right of the photo's title and choose **Delete**. A dialog box asks you to confirm.
4. Click **OK**.

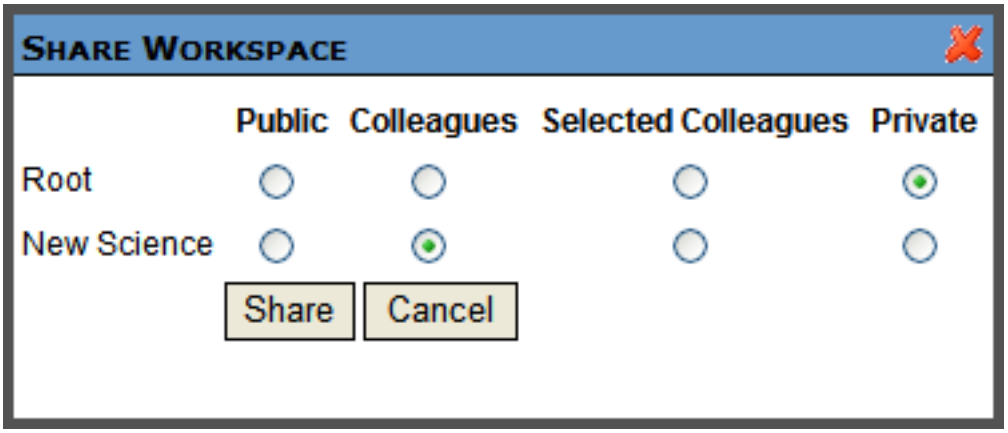
Sharing photos

The Photo Gallery lets you share photos with your colleagues. You can share photos with the Public, Colleagues, Selected Colleagues, or keep them private. You apply sharing options to categories, not individual photos. See also: *Public. Anyone who can access your profile* on page 2308

1. On the website, navigate to a user's **Profile Page > Photo Gallery**.
2. Click **Manage Folder** ( ). The Add box appears.



3. Click **Share Folder** ( ). The Share Workspace box appears.



4. Select with whom to share your photos.
   - **Public**. Anyone who can access your profile
   - **Colleagues**. Only colleagues (either you accepted their invitation, or they accepted yours)
   - **Selected Colleagues**. Colleagues that you designated as *selected*. See also: *Designating a selected colleague* on page 2266.
   - **Private**. Only you
5. Click **Share**.

Saving a photo to your local system

1. On the website, navigate to a **User's Profile Page > Photo Gallery**.
2. Select the category that contains the photo you want to save.
3. Click the triangle to the right of the photo's title and choose **Save As...** A dialog box asks you to open or save the file.
4. Click **Save**.

SocialBar

8.50 and higher

The SocialBar server control lets users of a community website bookmark colleagues, community groups, and content.

Inserting the SocialBar server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

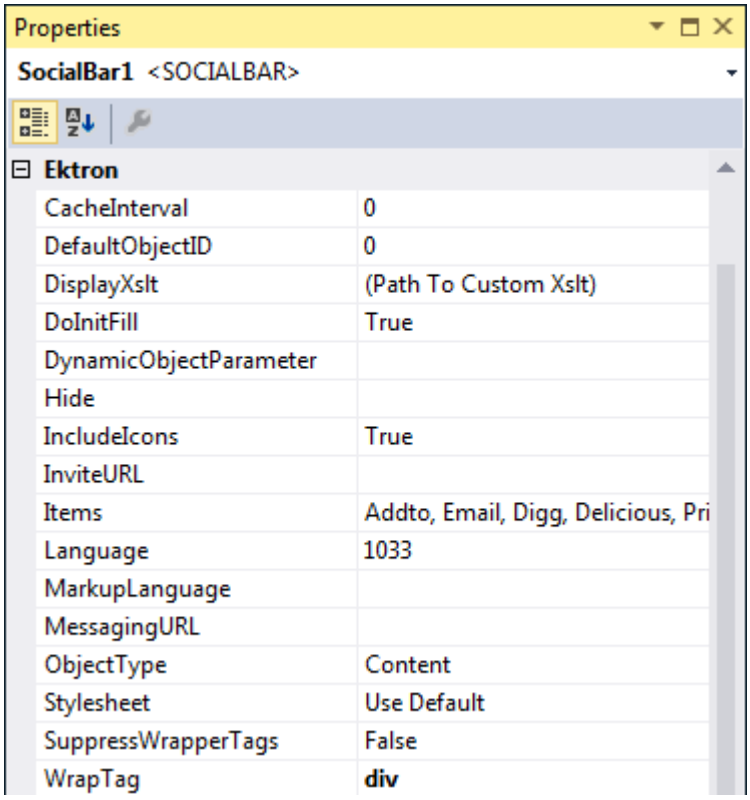
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **SocialBar** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:SocialBar ID="SocialBar1" runat="server" />
   ```

4. Click on `SocialBar` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



SocialBar properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultObjectID** (Long)

  The default object ID for this control to use when there is no matching dynamic parameter value passed. If you set this property set to 0 (zero) and leave the DynamicObjectParameter blank, the social bar can be used to add Web pages to your favorites.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicObjectParameter** (String)
  Gets or sets the QueryString parameter to read a object ID dynamically. To use the default object ID, leave blank.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **IncludeIcons** (Boolean)
  Select whether icons are displayed next to each item.
  - **True**. Display icons next to each item.
  - **False**. Do not display icons next to each item.
- **InviteURL** (String)
  The URL to the Invite server control's Web page. Two variables used within the URL.
  - **{0}**. Object ID
  - **{1}**. Object type

  The link must have both variables. The Web form can be relative or absolute. Below is an example.

```
invitepage.aspx?id={0}&type={1}
```

- **Items**(String)
  A comma-separated list of items that you want to appear on the Social Bar.
  - Add to.
    - If this control is associated with a *user*, the Addto item appears as:
      - **Add a Colleague**. If you are viewing another user's profile page and can add them as a colleague.
      - **CancelColleague Request**. If you are viewing another user's profile page who you asked to be a colleague but the user has not yet accepted.
      - **Remove From Colleagues**. If you are viewing a current colleague's profile page.
    - If the control is associated with a *community group*, the Addto item appears as:
      - **Join Group**. If you visit a community group that you have not joined.
      - **Leave Group**. If you already belong to a community group.

        - **CancelRequest to Join**. If you have tried join a restricted group and have not yet been accepted.
    - If the control is associated with *content*, the Addto item appears as:
        - **Add to Favorites**. If you view content that is not in your Favorites.
        - **Remove Favorites**. If you view content that is in your Favorites.
  - **Invite**. A link to the Web page that contains the Invite server control. This control lets you invite people to register on the site and become a colleague.

    IMPORTANT: For the Invite item to be active, you must enter a link to the page hosting the Invite control into the `InviteURL` property.

  - **GroupInvite**. Opens a dialog that invites colleagues or people who are not registered on the site to join the group.
  - **Email**. Email this Web page. Clicking this item launches your email with the subject and body loaded with the information about the content, user or community group.
  - **Digg**. Appears as **Digg It**. Launches `Digg.com`, a social bookmarking service.
  - **Delicious**. Launches `https://del.icio.us`, a social bookmarking service.
  - **Facebook**. Appears as **Facebook**. Launches Facebook, a social network service.
  - **Google**. Appears as **Google It**. Launches Google's social bookmarking service.
  - **Furl**. Appears as **Furl It**. Launches Furl, a social bookmarking service.
  - **Technorati**. Appears as **Technorati**. Launches Technorati, a social bookmarking service.
  - **Twitter**. Launches Twitter, a social network service, which lets you communicate with other Twitter users. If the Twitter link is clicked, and you log into a Twitter account, a link to the current page is inserted into the Status box.
  - **Yahoo**. Appears as **Yahoo! It**. Launches Yahoo's social bookmarking service.
  - **Print**. Print the Web page.
  - **PrivateMessageUser**. Creates a link that lets you send a Private Message to a user. See also:
  - **PrivateMessageAdmin**. Creates a link that lets you send a Private Message to a community group's administrator. See also: .

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **MarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.See also: *Ektron Markup Language* on page 2633*Ektron Markup Language* on page 2633
- **MessagingURL** (String)
  The Web page that hosts the Messaging server control. This property is used when either `PrivateMessageUser` or `PrivateMessageAdmin` appears in the **Items** property.

  If a value appears in the `ek_RedirectFromLoginKeyName` key in the `web.config` file, the user is returned to the original URL after sending the message. By default, this value is `RedirectUrl`. If you remove this value and do not add another, the person sending a message sees a note stating "Your message has been sent."
- **ObjectType**
  The type of object to which this control is assigned. Choices are:
  - Content
  - User
  - Group
  - **Data Type**. Ektron.Cms.Common.EkEnumeration.CMSSocialBarType**s**
- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.

- **Div**. Apply attributes to a block of code.
- **Custom**. Lets you use a custom tag.

Using the SocialBar server control

The SocialBar server control lets users of a community website bookmark colleagues, community groups, and content. The following table explains the details.

Item being bookmarked	Description
Users	• Sends colleague requests • Removes colleagues when a user visits another user's profile page
Group	• Join and leave community group • Invite others to join group
Content	Add or remove content and URLs from a user's Favorites. See also: *Favorites* on page 2254

The Social Bar control also is used to bookmark Web pages with Social Bookmarking services, such as Digg.com or del.icio.us. By default, Digg.com and del.icio.us are loaded in the `Items` property of the server control.

In addition, the Social Bar control can print or email a Web page, and invite non-site users to become members of the site. Clicking **Invite** links to a Web page that hosts the Invite server control. See also: *Invite* on page 2267.

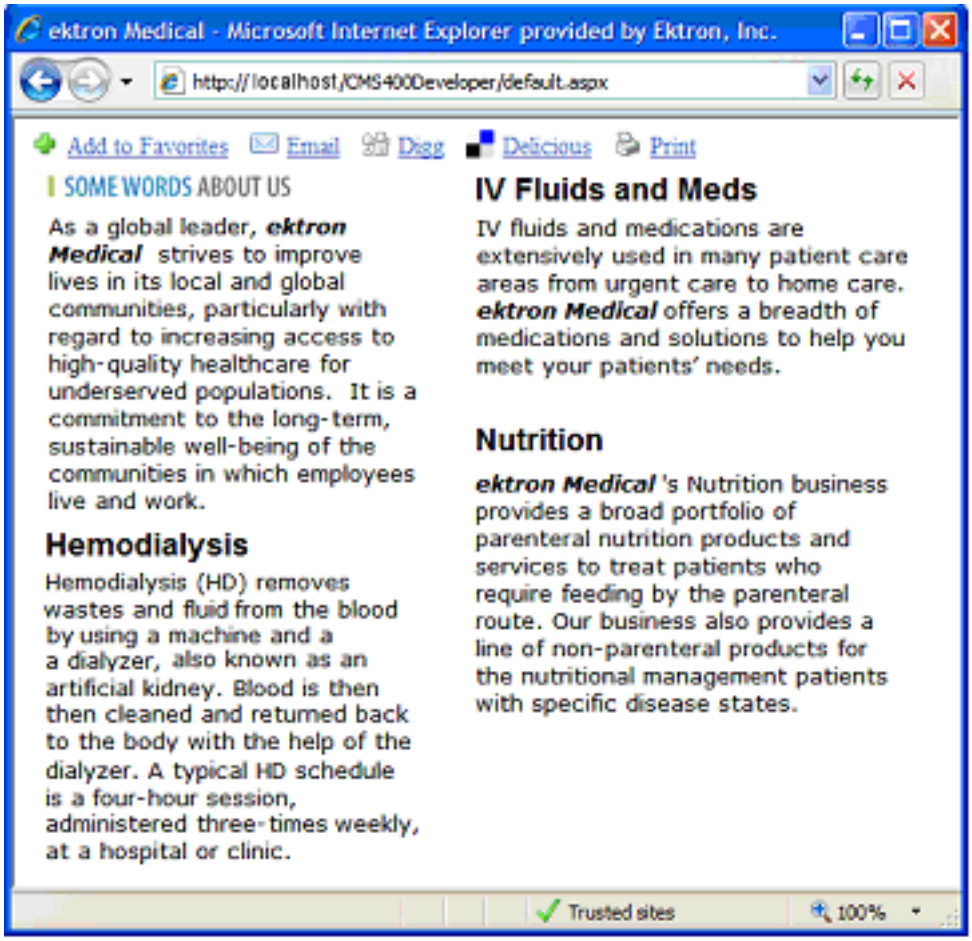
Adding a Web page URL to a favorites via the social bar

You can let users add a Web page's URL to their Favorites when they click **Add to Favorites** on the Social Bar. Typically, you would do this if more than one content item appears on a Web page, but you want only one Social Bar control on the page.

1. Drag and drop a SocialBar server control on a Web form.
2. Set the `DefaultObjectID` property to **0** (zero).
3. Leave the `DynamicObjectParameter` property blank.
4. If it is not already there, copy the comma-separated list of items from the `Items` property to the **Addto** item property.
5. Set the `ObjectType` property to **Content**.

The image below shows a Web page with several content items. A Social Bar control is associated with each one. In this example, when a you click **Add to Favorites**, you add the associated *content item* to your Favorites.

In the second image, one Social Bar appears on the Web page with several content items. In this example, if you click **Add to Favorites** on the Social Bar, you add the Web page's *URL* to your Favorites.



See also: *Adding a URL to your favorites* on page 2258

Sending a private message from the social bar

You can send private messages to another user or the community group administrator from the Social Bar control.

1. Drop a SocialBar server control on a user's profile or community group's page.
2. Set the `ObjectType` property to User or Group.
3. Add either **PrivateMessageUser** or **PrivateMessageAdmin** to `Items` property.
   - Use `PrivateMessageUser` to let a user send a message to the user whose profile is being viewed.
   - Use `PrivateMessageAdmin` to let a user send a message to the group administrator.
4. In the `MessagingUrl` property, identify a page that contains the Messaging server control. See also: *Messaging* on page 2276.
5. Make sure the `DynamicObjectParameter` is set, if necessary.

Tweeting the current URL

You can let site visitors use Twitter to tweet their current browsing location by adding the Twitter item to the list of parameters in the SocialBar's `Items` property.

After the item is added, a Twitter icon and link appear in the Social Bar. When you click the link, Twitter is launched. After you log into your Twitter account, your current browsing location appears in the "What are you doing?" status box. You then can click Twitter's **Update** button to post the status.

1. Drag and drop a SocialBar server control on a Web form.
2. Make sure the `DefaultObjectID` property is set to **0** (zero).
3. Leave the `DynamicObjectParameter` property blank.
4. If it is not already there, add the **Twitter** item to the comma-separated list in the `Items` property.
5. Set the `ObjectType` property to **Content**, **User** or **Group**, depending on the type of page you are editing.

TagCloud

8.50 and higher

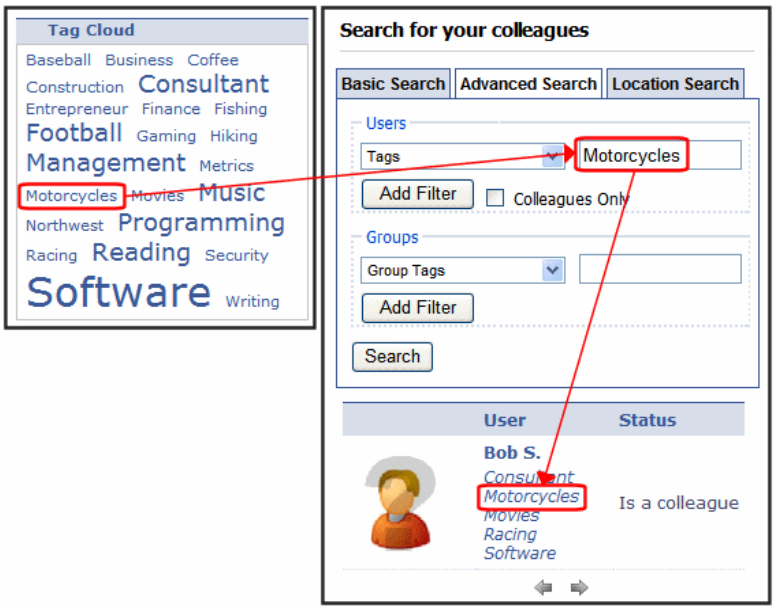
The TagCloud server control shows a weighted list of tags assigned to users, community groups, content, or library items. Tag sizes are proportional to the number of times they are assigned. For example, there are 2 tags, *Software* and *Programming*. If the *Software* tag is used 5 times and the *Programming* tag is used 2 times, the *Software* tag is approximately twice as large.

Use the `TagType` property to determine which types of tags appear in the cloud. For example, set `TagType` to `User` to display user tags.

You can set a maximum number of tags to display via the `MaxTagsDisplayed` property. This property makes sense if the `Orderby` property is set to `Taggedcount`. In this case, the cloud only displays items with the highest number of tags.

Use the `OrderBy` property to sort tags within a cloud. Your choices are `alphabetical` or by `Taggedcount` (the number of times an item is tagged). Then, use the `OrderByDirection` property to determine if items are sorted in ascending or descending order.

Within a tag cloud, you can link tags to their source items. If you do, site visitors can click a tag to launch a search of all users, community groups, content, or library items to which the tag is assigned. For example, if someone views a user-based tag cloud and clicks **Motorcycles**, the community search page appears, showing all users tagged with Motorcycles.



To make a tag cloud's items searchable, add a path to the Web form containing the Search server control in the `TagTemplate` property.

Inserting the TagCloud server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

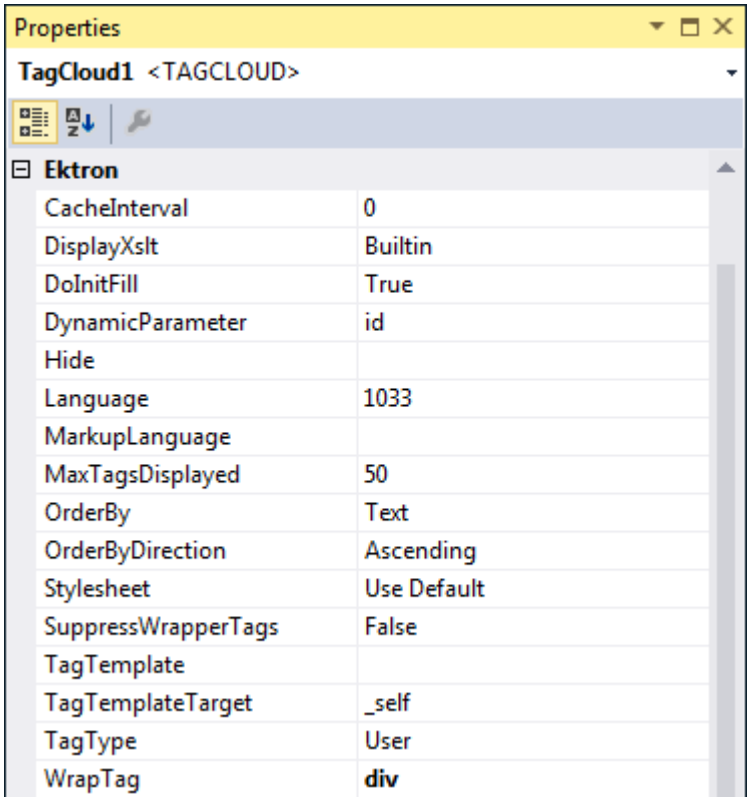
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **TagCloud** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:TagCloud ID="TagCloud1" runat="server" />
```

4. Click on `TagCloud` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



TagCloud properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)
  By default, the `TagTemplate` property passes a `TagId` parameter, whose value is the tag that a site visitor clicked in the tag cloud. Use this property to manually override that parameter and set the TagID value by hand.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)(missing snippet link)
- **MarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.See also: *Ektron Markup Language* on page 2633*Ektron Markup Language* on page 2633
- **MaxTagsDisplayed** (Integer)
  The maximum number of tags to display. 0 (zero) = unlimited. If you set a maximum, and more than that number of tags are applied, then only the most frequently-used tags appear. For example, if you enter ten, the ten tags applied the most number of times appear.
- **OrderBy** (String)
  The criteria by which tags are ordered in the tag cloud. Choices are:
  - **TaggedCount**. Sort by how many times a tag is assigned
  - **Text**. Sort alphabetically by tag name
- **OrderByDirection** (String)
  Select the direction of the `OrderBy` property. Choose Ascending or Descending.
- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.

- **True**. Suppress wrap tags.
- **False** (default). Allow wrap tags.

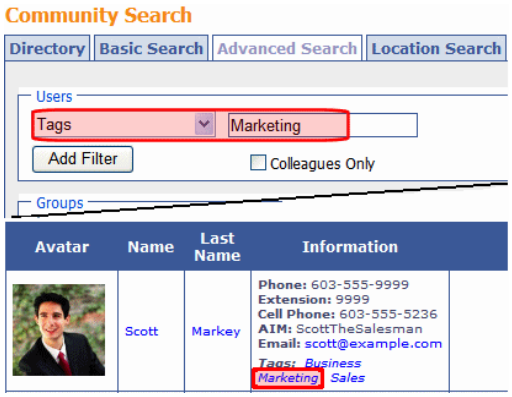
- **TagTemplate** (String)

  If you want a tag cloud's items to link to a page that shows *all* items with that tag, add a path to a Web form page that hosts the appropriate search control. The path can be relative or absolute.

  - If the `TagType` is `User` or `Community Group`, enter a page hosting a Community Search server control.
  - If the `TagType` is `Content` or `Library`, enter a page hosting a templated search server control.

  For example, in the Ektron Intranet starter site's `tagcloud.aspx` page, the first 2 TagCloud controls link to users and community groups. So, this property is set to `CommunitySearch.aspx`, a Web form that hosts the CommunitySearch server control.

  When a site visitor clicks a tag cloud item, the Community Search page displays, populated with search results for the clicked tag. The visitor can click any result to see more about it.



  The following 5 parameters are added to the link's QueryString, and pass tag information to the destination Web form. Its search server control uses the parameters to populate the search.

  - **searchtag** or **searchgrptag**. Searchtag represents the tag's text for users; searchgrptag represent the tag's text for community groups
  - **TagId**. The clicked tag's ID
  - **TagLanguage**. The clicked tag's language; only search results in this language appear
  - **TagCount**. The number of times the tag has been assigned
  - **TagType**. The tag's type: user, community group, content or library item. This value is obtained from the `TagType` property (described below).

  In addition to these parameters, you can add your own by defining them in the path. If you do, the above 5 parameters are appended to yours.

- **TagTemplateTarget** (ItemLinkTargets)
  Determines the type of window that appears when you click a link in the server control.
  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **TagType** (String)
  Select the type of tags that appear in the tag cloud. Choices are:
  - User
  - CommunityGroup
  - Content
  - Library
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

UserProfile

8.50 and higher

A user's profile page is the person's home on the website. The UserProfile control displays following information about a user.

- Avatar
- Personal tags
- Screen name
- Custom properties
- If community aliasing for users is enabled, the user's alias (following **Profile**

Links)

Logged-in users can edit their profile by clicking **Edit Profile** in the top right corner. This dialog box is the same dialog that a membership user uses to create an account on the site. See Membership Users and Groups for a description of this dialog. To automatically create a friendly URL for a user profile, you can use Community Aliasing for Users. See also: Types of URL Aliasing.

Inserting the UserProfile server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

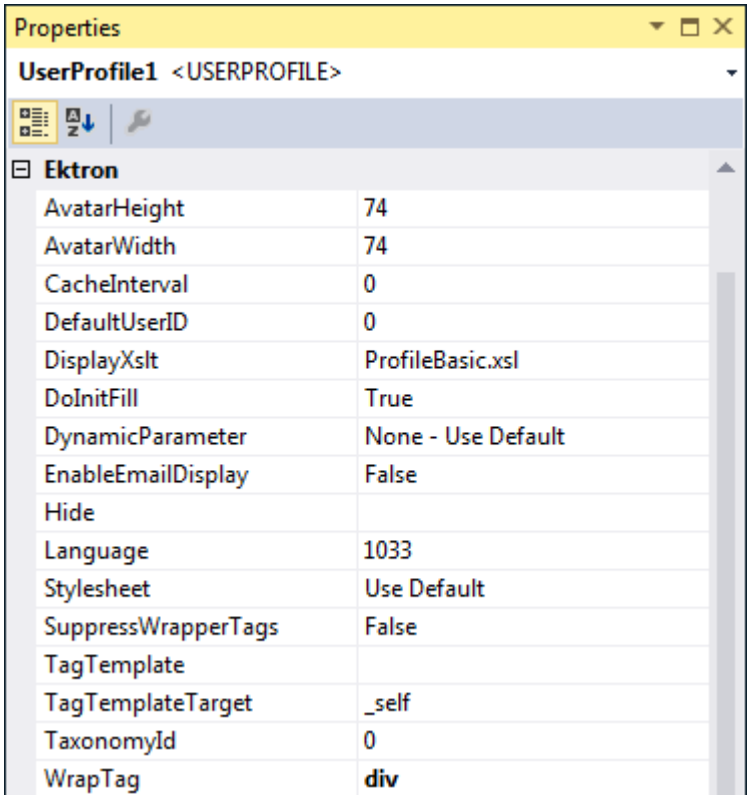
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **UserProfile** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:UserProfile ID="UserProfile1" runat="server" />
```

4. Click on `UserProfile` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



UserProfile properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AvatarHeight** (Integer)
  The display height in pixels of the avatar in the profile area.
- **AvatarWidth** (Integer)
  The display width in pixels of the avatar in the profile area.
- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **DefaultUserID** (Long)
  The default user ID for this control to use when there is no matching dynamic parameter value passed.
- **DisplayXslt** (String)
  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a user ID dynamically. To use the default object ID, leave blank.

- **EnableEmailDisplay** (Boolean)

  Set to True to display a user's email address in the profile. If a user's **Private Profile** setting is set to `Private`, the profile information is not visible, regardless of this property's setting. If **Private Profile** is set to `Colleagues`, only a user's colleagues can see email information.

  - **True**. Display a user's email address in the profile.
  - **False**. Do not display a user's email address in the profile.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

- **TagTemplate** (String)

  The Web page template that contains the CommunitySearch server control. This allows users to search for others users with the same tags. If a template is entered, Personal Tags in the Profile appear as links. Clicking a tag forwards the user to a User Search page that displays search results for the matching tag.

- **TagTemplateTarget** (ItemLinkTargets)
  Determines the type of window that appears when you click a link in the server control.
  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **TaxonomyId** (Long)
  The numeric ID of the taxonomy that is available to users. A user editing a profile can select which categories to associate with their profiles. See also: Organizing Content with Taxonomies
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Content server controls

8.50 and higher

The Content server controls are as follows:

- *ContentBlock* below
- *ContentList* on page 2336
- *ContentReview* on page 2342

ContentBlock

8.50 and higher

IMPORTANT: Starting from release 8.6, the ContentBlock server control was replaced by the FrameworkUI: <ektron:ContentView> templated server control. If you are already using the ContentBlock server control, you can continue to do so, but Ektron recommends using current versions of functionality.

The ContentBlock server control displays the following types of content blocks on an Web page.

- **Static**. Displays one specified content block
- **Dynamic**. Displays the content block of the ID passed through a URL parameter
- **XML**. Displays the content from the XML or XHTML code

Inserting the ContentBlock server control onto a page

PREREQUISITE

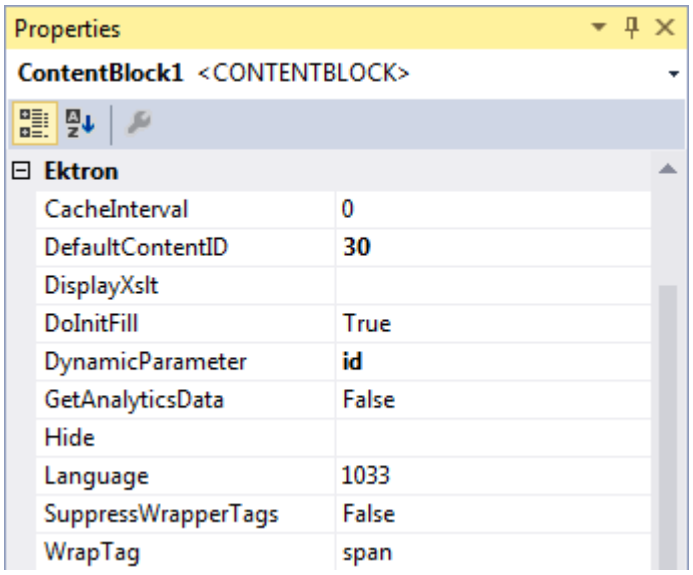
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **ContentBlock** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:ContentBlock ID="ContentBlock1" runat="server" />
```

4. Click on `ContentBlock` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



ContentBlock properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultContentID** (Long)

  The ID of a content block that appears where you insert this server control. If you don't know the ID number of the content block, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147.

NOTE: If you identify a content block that displays an Office document which will be published as HTML, make sure the template sets `<span>` tags to display the content as a block.

- **DisplayXslt** (String)
  Ignore for a non-XML content block.
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)
  Select **None - Use Default**. This parameter is used for dynamic content blocks.
- **GetAnalyticsData** (Boolean)
  Set this property to **True** if you want the following information for each content in the list. Returns **Content View Count**, **Content Rating**, **Content Rating Average**. Create your own XSLT styles to display this data.

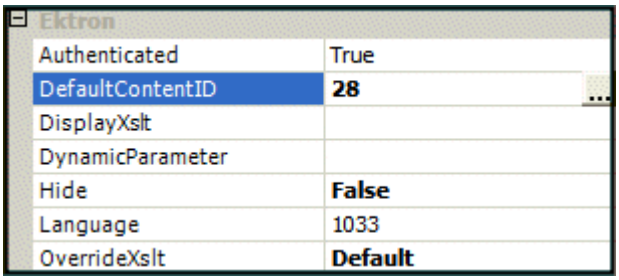
  IMPORTANT: This property provides reliable data only when the Business Analytics Feature is on. See Running Ektron Business Analytics.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)

  IMPORTANT: If you want to let content authors edit this content using Ektron's Edit in Context feature, this must be set to false.

  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

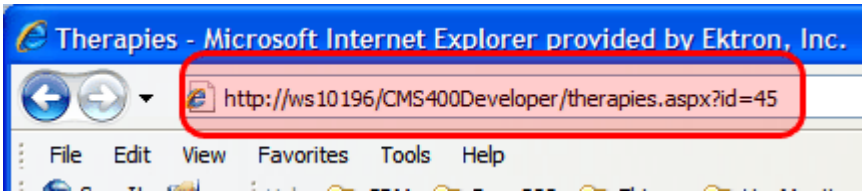
Using a static content block

A static content block displays one identified content block on a Web page. The following image shows a server control that retrieves content block ID=28 and displays it in the browser.

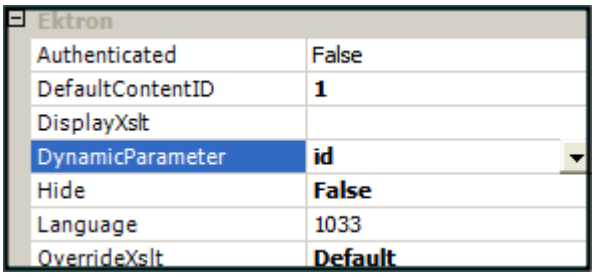


Using a dynamic content block

Use a dynamic content block to display a content block whose ID is passed through a URL parameter. You would use this server control with a dynamic template.



The following example uses the ContentBlock server control to pass the id as a URL parameter. When a user clicks a link that passes the content block ID as a URL parameter, that content block appears. If that content block is not available, content block 1 appears



The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

Dynamic ContentBlock properties

- **Authenticated** (String)

  Indicates if you are logged into the CMS Explorer and can use it to browse to content, collections, and so on. See also: *Browsing your Ektron site using CMS Explorer* on page 2147.

- **DefaultContentID** (Long)

  The ID of a content block that appears where you insert this server control. If you don't know the ID number of the content block, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

  NOTE: If you identify a content block that displays an Office document which will be published as HTML, make sure the template sets `<span>` tags to display the content as a block.

- **DisplayXslt** (String)

  Ignore for a non-XML content block.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Select **id**. When you do, this server control uses the content block passed as a URL parameter.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

  IMPORTANT: If you want to let content authors edit this content using Ektron's Edit in Context feature, this must be set to false.

- **WrapTag** (String)

  Lets a developer specify a server control's tag.

  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

XML ContentBlock properties

Use an XML content block to display an XML content block on an Ektron Web page.

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **Authenticated** (String)

  Indicates if you are logged into the CMS Explorer and can use it to browse to content, collections, and so on. See also: *Browsing your Ektron site using CMS Explorer* on page 2147.

- **CacheInterval** (Double)

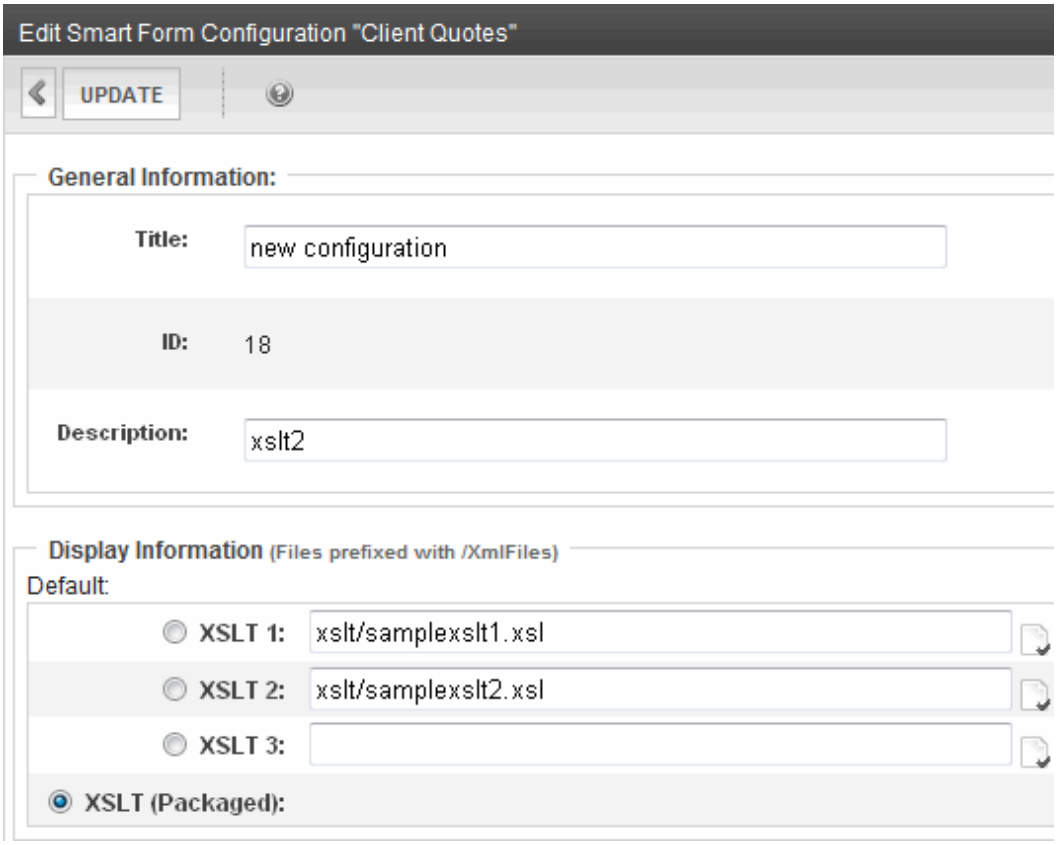
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultContentID** (Long)

  The ID of a content block that appears where you inserted this server control if no other content block is identified, or is not available. If you don't know the ID number of the content block, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page. If you do not define the `DisplayXslt` property, the `OverrideXslt` property specifies an XSLT identified in the Edit Smart Form Configuration screen. See also: Working with Smart Forms, and the Ektron KnowledgeBase article "HOW TO:Add acceptable XSLT files to the system".



WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)
  To make this content block dynamic, select **id**. When you do, this server control uses the content block passed as a URL parameter.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Retrieving the XML structure of an XML ContentBlock

Retrieving the XML structure of XML content allows for greater control over developing XSLs. The following example shows how to retrieve the XML structure.

1. Open a new Web form.
2. Drag and drop a Content Block server control onto it.
3. Set the DefaultContentID to an XML content block.

IMPORTANT: This does not work with HTML content blocks, as there is no XML structure to output.

4. Drag and drop a textbox on the Web form.

5. Set the `TextMode` property to MultiLine.
6. It is also recommended that you set the width of the text box to at least 400 px.

   NOTE: On the code-behind page, add the following line.
   `TextBox1.Text = ContentBlock1.EkItem.Html`

7. Build the project.
8. View the form in a browser. The XML structure of the content block appears in the text box.

Using ContentBlock programmatically

The following code displays a content block:

NOTE: Before adding these lines of code, drag and drop a literal box on your Web form.

```
Dim MyContentBlock As New ContentBlock
  MyContentBlock.DefaultContentID = 8
  MyContentBlock.Page = Page
  MyContentBlock.Fill()
  Literal1.Text = MyContentBlock.EkItem.Html
```

To display a content block with the content block title:

NOTE: Before adding these lines of code, drag and drop a 2 literal boxes on your Web form.

```
Dim MyContentBlock As New ContentBlock
  MyContentBlock.DefaultContentID = 8
  MyContentBlock.Page = Page
  MyContentBlock.Fill()
  Literal1.Text = MyContentBlock.EkItem.Title
  Literal2.Text = MyContentBlock.EkItem.Html
```

ContentList

8.50 and higher

IMPORTANT: Starting from release 8.6, the ContentList server control was replaced by the FrameworkUI: <ektron:ContentView> templated server control. If you are already using the ContentList server control, you can continue to do so, but Ektron recommends using current versions of functionality.

The ContentList server control displays a list of content blocks on a Web page. In contrast to a List Summary, where content must be in a specified folder, the ContentList server control displays content from any Ektron folder. Depending on the setting you choose for `DisplayXslt`, you can change information displayed for each content block.

When added to a template and visited, a ContentList Summary looks similar to the following.

The ContentList server control has the following options to display a content list.

- Define a content list in a content block's metadata. Then, assign that content block's ID in the `DefaultContentID` property.

  When using this option, an administrator typically sets up the Workarea portion of the process. Then, a developer adds the server control to a Web form and assigns the content block ID and the metadata name to the appropriate properties. Because the content list is assigned to a content's metadata, you can pass the content ID dynamically in a Web form and display a list for each content block you defined.

- Assign a comma-delimited list of content blocks to the `ContentIds` property. See also:

  Using the `ContentIds` process, a developer adds the ContentList server control to Web form. Then, the developer defines a list of content IDs in the `ContentIds` property. If the server control or the list in the `ContentIds` property is deleted, it is not available and will have to be created again.

Inserting the ContentList server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

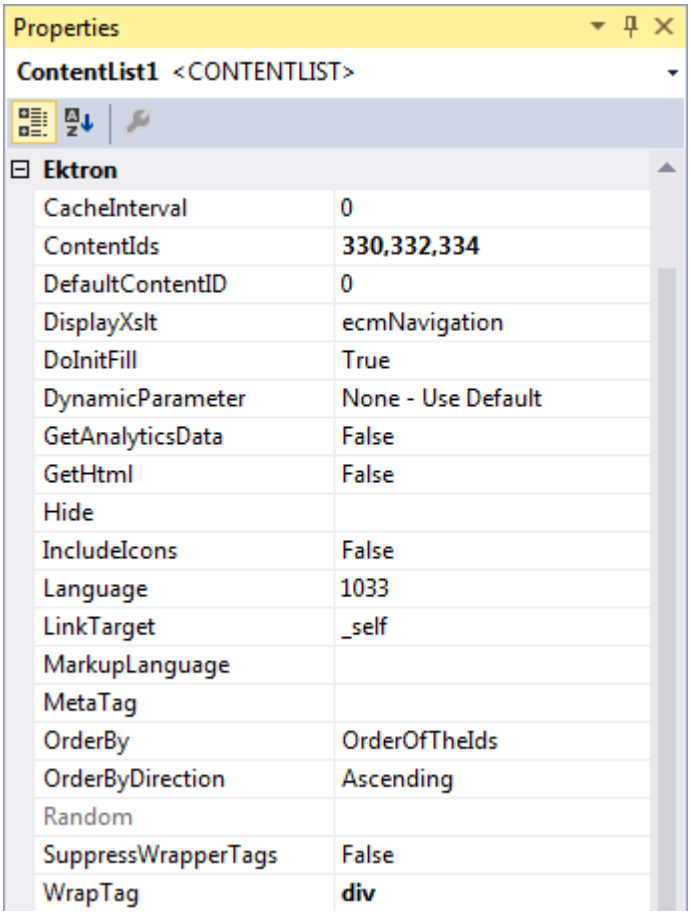
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **ContentList** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:ContentList ID="ContentList1" runat="server" />
   ```

4. Click on `ContentList` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



ContentList properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **ContentIds** (String)
  A comma delimited list of content block IDs.
- **DefaultContentID** (Long)
  Set content ID value. When set, content IDs are generated from the MetaTag value for this content.
- **DisplayXslt** (String)
  Determines how information appears on the page.
  - **None**. Databind only
  - **ecmNavigation**. Lists the title of each content block.

- **ecmTeaser**. Lists the title of each content block plus the content summary.
- **ecmUnOrderedList**. Sorts the list in no particular order. Shows the title and content summary.
- **Path to Custom Xslt**. If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

WARNING! If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Uses the QueryString parameter to read a content ID dynamically.

  - **None - Use Default**. Use the default content ID list.
  - **ID**. Reads a content block's ID dynamically.
  - **ekfrm**. Reads a form block's ID dynamically.

- **GetAnalyticsData** (Boolean)

  Set this property to **True** if you want the following information for each content in the list. Returns **Content View Count**, **Content Rating**, **Content Rating Average**. Create your own XSLT styles to display this data.

IMPORTANT: This property provides reliable data only when the Business Analytics Feature is on. See Running Ektron Business Analytics.

- **GetHtml** (Boolean)
  - **True**. Retrieve and display content (html body) for all content blocks in the list summary. For example, to display content inside a Web server control such as a GridView.
  - **False**. Do not get and display HTML.
- **Hide** (Boolean)

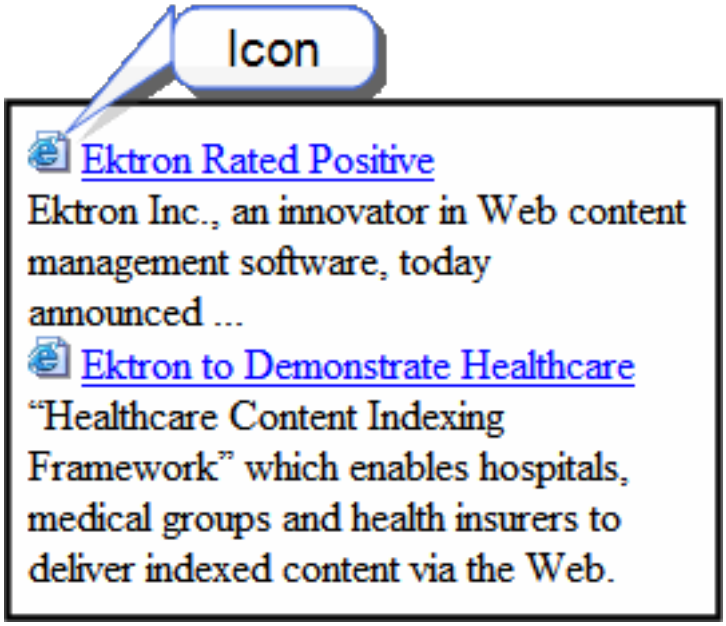
  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **IncludeIcons** (Boolean)

  Choose whether to display icons next to the content list's links.

IMPORTANT: This property works only when ecmSummary or ecmTeaser are used in the `DisplayXslt` property. When the `[$ImageIcon]` variable is used in an EkML file and that file is assigned to the `MarkupLanguage` property, this property acts as True.



- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LinkTarget** (ItemLinkTargets)

  Determines the type of window that appears when you click a link in the server control.

  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.

- **MarkupLanguage** (String)

  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.See also: *Ektron Markup Language* on page 2633

  See also: *contentlist.ekml* on page 2635

NOTE: If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored. If the EkML file contains the `[$ImageIcon]` variable, the `IncludeIcons` property acts as True.

- **MetaTag** (String)

  Specify a Metadata definition whose type is Content Selector. When you do, the associated list of content items will appear where you place the server control.

  NOTE: You cannot insert other metadata types.

  This works with the `DefaultContentID` property. For more information about using metadata to assign a list of related content to a content item, see Creating and Deploying a Related Content Definition.

- **OrderBy** (Ektron.Cms.Controls.CmsWebService.ContentListOrderBy)

  Sort the list by one of these values:

  - **Title**. Alphabetically, by content title
  - **DateModified**. The last date content was modified
  - **DateCreated**. The date content was created
  - **LastEditorFname**. The last editor's first name
  - **LastEditorLname**. The last editor's last name
  - **OrderOfTheIds**. Preserves content ID order based on `ContentIds` property
  - **ContentRatingAverage**. Business Analytics Content Rating
  - **ContentViewCount**. Business Analytics Content Views

- **OrderByDirection** (Ektron.Cms.Controls.CmsWebService.ContentListOrderByDirection)

  Determines which direction to sort content determined by the `OrderBy` property.

  - **ascending**. Items are arranged A, B, C or 1,2,3.
  - **descending**. Items are arranged. Z,Y,X or 3,2,1.

  If sorting by date, descending puts the most recent first. When `ascending` is selected and the `OrderBy` property is set to `OrderOfTheIds`, the order of the IDs are preserved. When set to descending, the order is reversed.

- **Random** (Boolean)
  - **True**. Randomly display one content block link from the content list. The content changes each time a user views the page.
  - **False**. Display the content list normally.

  NOTE: If you use a custom XSLT or EkML file, the type of content displayed can be manipulated. For example, if you use an EkML file that has the [$Html] variable in it, the actual content appears instead of a link. See also: *Ektron Markup Language* on page 2633 and *[$Html]* on page 2666

- **SuppressWrapperTags** (Boolean)

  This property is set to `false` because Ajax uses `<div>` tags to rewrite the region around the tag. You *cannot* change the value to `true`.

- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

ContentReview

8.50 and higher

The ContentReview server control lets site visitors rate content on your site. Place this control on a template that displays content items or eCommerce products. For example, you place this control on a Master page and set its `DynamicParameter` property to ID. Then, when a Web form containing a content item or product passes its ID to the QueryString, a site visitor can use the control to record a review.

You can use the ContentReview server control as follows:

- Place a star-based scale on a Web page. Site visitors use the scale to rate a content item. Depending on the XSLT, they can also submit review comments.
- Display reviews and comments about a content item, or by a specific site or membership user.

See also: User-Ranking of Content.

NOTE: On a PageBuilder page, you can insert a ContentReview server control using the Calendar widget. See also: Creating and Using Widgets. The eCommerce Product List server control also can display the star-based scale.

Inserting the ContentReview server control onto a page

PREREQUISITE

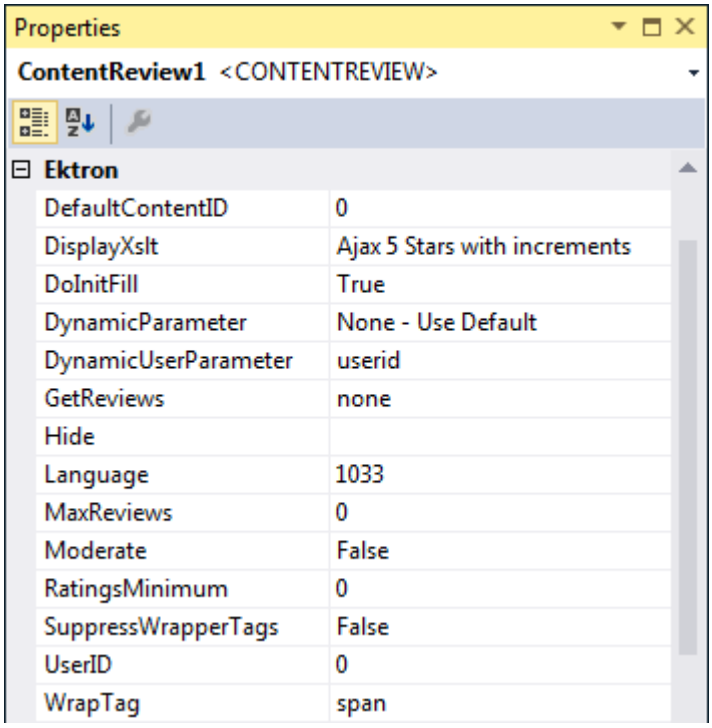
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **ContentReview** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:ContentReview ID="ContentReview1" runat="server" />
   ```
4. Click on `ContentReview` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as

you modify the property values.



ContentReview properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **DefaultContentID** (Long)

  The ID of a content block being rated by this server control. If you don't know the ID number of the content block, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

- **DisplayXslt** (String)

  Select the type of review to display, or enter the path to a custom XSLT. For additional information, see *Using the DisplayXSLT property* on page 2345. Choices are:

  - **Ajax 5 Stars**. A 5-star rating system utilizing Ajax for display.
  - **Ajax 5 Stars Comment**. A 5-star rating system utilizing Ajax for display. When you hover over the stars, a comment box appears. Comments that are added are submitted via an Ajax call.
  - **Ajax 5 Stars with Increments**. A 5-star rating system with half star increments that utilizes Ajax for display.
  - **5 Stars**. A 5-star rating system that allows visitors to add text reviews of content.

- **5 Stars with Increments**. A 5-star rating system with half star increments that allows visitors to rate and add a text review of the page's content.
- **Review List. D**isplays a list of reviews for content or a user.
- **Path to Custom Xslt**. If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  To make this content review control dynamic, select **id**. When you do, this server control is attached to the content block passed as a URL parameter.

- **DynamicUserParameter** (String)

  When using this control to retrieve a user's reviews, set this property to **UserId** to make the user's ID dynamic.

- **GetReviews** (ReviewTypes)

  Returns a list of reviews for content or a user.

  - **None**. Do not return reviews.
  - **Content**. Returns reviews based on the content ID provided in the `DefaultContentID` property.
  - **User**. Returns reviews based on the User ID provided in the `UserId` property.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **MaxReviews** (Integer)

  The number of reviews to retrieve if the `GetReview` property is set to Content or User. 0 (zero) = unlimited.

- **Moderate** (Boolean)
  Setting this property to True allows Ektron users to moderate reviews. See also: Moderating Reviews.
- **RatingsMinimum** (Integer)
  Sets a minimum number of reviews before displaying the average rating. When set to 0 (zero), the average rating appears as soon as content is rated.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **UserID** (Long)
  The user ID for which to get reviews. If left blank, reviews from all users are returned. If a user ID is specified, only reviews for that user are returned.

  IMPORTANT: The `GetReviews` property must be set to User for the control to use this property.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Using the DisplayXSLT property

Reviews communicate your site community's feelings about a product or article. They also give community members a voice. The ContentReview server control's `DisplayXSLT` property manages the display of reviews on your site. The control can display reviews for specific content or a specific user.

The following list shows an example of each XSLT as it appears on a Web page with a description and the XSL file being used. The files are located in *webroot*`\`*siteroot*`\Workarea\Xslt`.

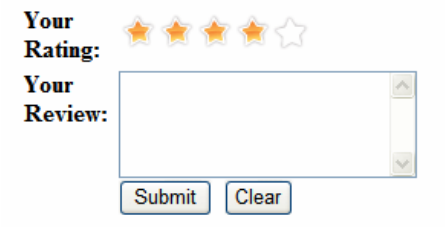
WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **Ajax 5 Stars**. `rating5star.xsl`. A 5-star rating system utilizing Ajax for display.

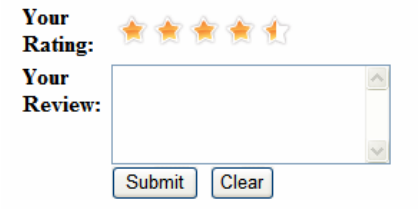
- **Ajax 5 Stars Comment**. `rating5starComment.xsl`. A 5-star rating system utilizing Ajax for display. When you hover over the stars, a review box pops-up. Visitor comments are submitted via an Ajax call.

- **Ajax 5 Stars with Increments**. `rating5starinc.xsl`. A 5-star rating system with half-star increments that uses Ajax for display.

- **5 Stars**. `rating5starAddEdit.xsl`. A 5-star rating system that lets visitors add text reviews of content.



- **5 Stars with Increments**. `rating5starincAddEdit.xsl`. A 5-star rating system with half star increments that allows visitors to add text reviews of content.



- **Review List**. `ratinglist.xsl`. Displays a list of reviews for content or a user.

Retrieving the XML structure of a ContentReview

Retrieving the XML structure of XML content allows for greater control over developing XSLs.

1. Open a new Web form.
2. Drag and drop a ContentReview server control onto it.
3. Set the `DefaultContentID` property.
4. Drag and drop a textbox on the Web form.
5. Set its `TextMode` property to **MultiLine**. You should set the width of the text box to at least 400px.

6. On the code-behind page, add the following line.
7. `Textbox1.Text = ContentReview.XmlDoc.InnerXml`
8. Build the project.
9. View the Web form in a browser. The XML structure of the collection appears in the textbox.

DesignTimeDiagnostic

8.50 and higher

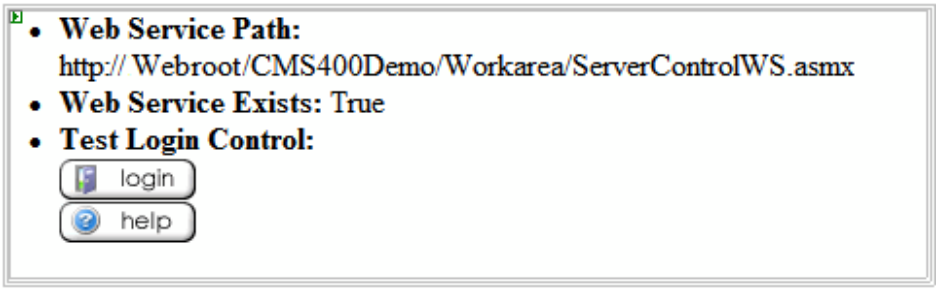
IMPORTANT: This control is for use in design-time only. Nothing is rendered at run-time.

The DesignTimeDiagnostic server control is used to verify the connection to Ektron's server controls Web services. When added to a Web form, this control provides the following information:

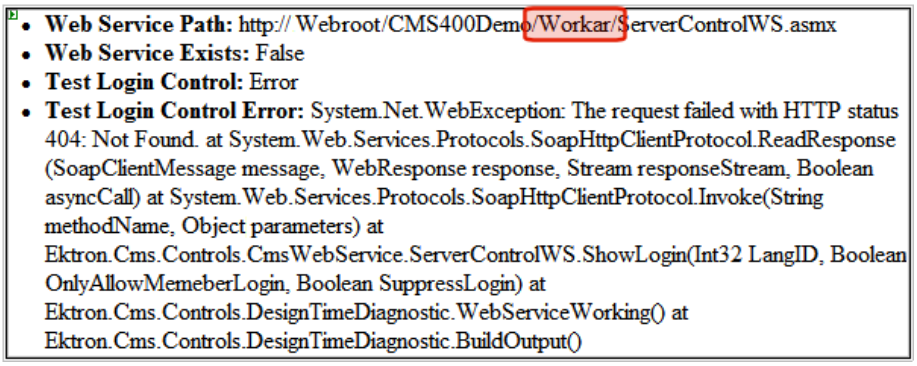
- **Web Service Path**. The server control Web service path in your web.config file.
- **Web Service Exists**. Calls a method in the Web service that returns **True** if the Web service exists. If it does not, it returns **False**.
- **Test Login Control**. Displays a Login server control to show that the Web service is connected and working properly. When not connecting properly, an error message appears.

The DesignTimeDiagnostic server control has no definable Ektron properties.

The following example shows the control on a page connected to the Web service.



The following example shows the control on a page not connecting to the Web service. Note that the path is not the correct path. It should be:
`http://192.168.0.82/siteroot/Workarea/ServerControlWS.asmx`



Inserting the DesignTimeDiagnostic server control onto a page

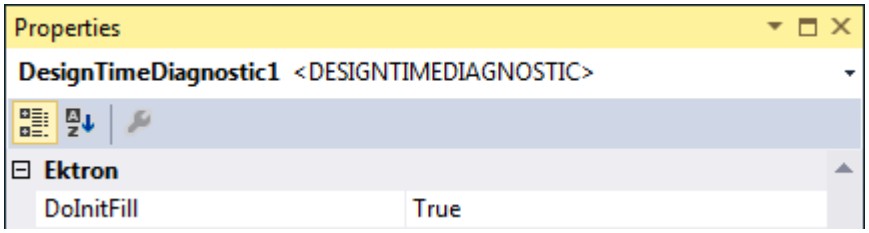
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **DesignTimeDiagnostic** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:DesignTimeDiagnostic ID="DesignTimeDiagnostic1" runat="server" />
```

4. Click on `DesignTimeDiagnostic` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



DoInitFill (Boolean)

By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

Directory

8.50 and higher

The Directory server control lets you customize the behavior of the taxonomy feature. You place this control on a Web form to display a taxonomy. See also: Organizing Content with Taxonomies.

NOTE: To display a taxonomy on a PageBuilder page, use the taxonomy summary widget.

To minimize taxonomy's impact on the performance of your production server, follow these guidelines.

- Use the default page size (50). If you need to customize page size, use Ektron's Developer Reference Sample SiteAPI, but do not increase it above 800.

- For large databases, do not increase the value of the `TaxonomyDepth` property above 1. Changing this value dramatically slows down your production server's performance. However, you can increase depth on your staging server for testing purposes.

Inserting the Directory server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

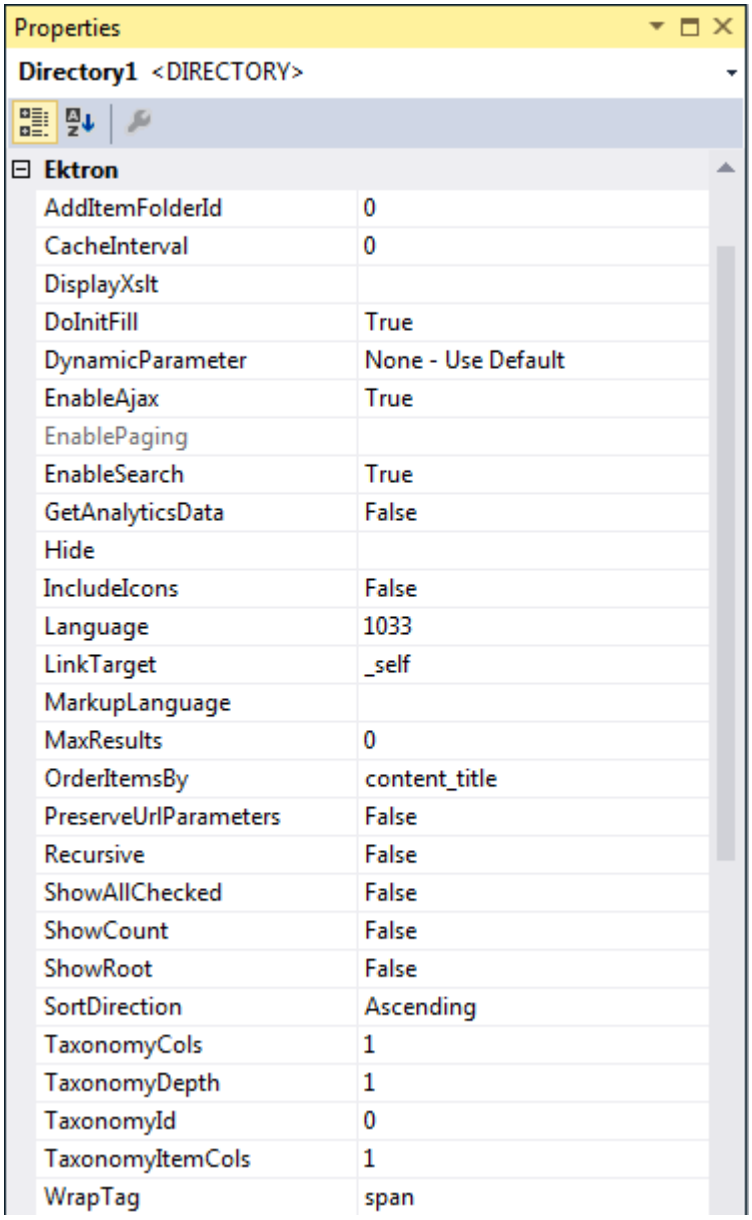
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Directory** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Directory ID="Directory1" runat="server" />
```

4. Click on `Directory` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



Directory properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AddItemFolderID** (Long)

  Define the Ektron folder ID where content is stored when an item is added to the taxonomy via the Add Asset link or the Add Content link. To enable adding content items via the Directory server control, you must use an EkML file with the `[$AddAsset]` and `[$AddArticle]` variables. See also: *taxonomy.ekml* on page 2641.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

  NOTE: If the `EnablePaging` property is set to True, the `CacheInterval` property is disabled.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

  WARNING!
  If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a content ID dynamically.

- **EnableAjax** (Boolean)

  Set to true to enable Ajax searches. When enabled, the `MaxResults` property determines the maximum number of results per page.

  - **True**. Enable Ajax Search (default value)
  - **False**. Original HTML Search

- **EnablePaging** (Boolean)

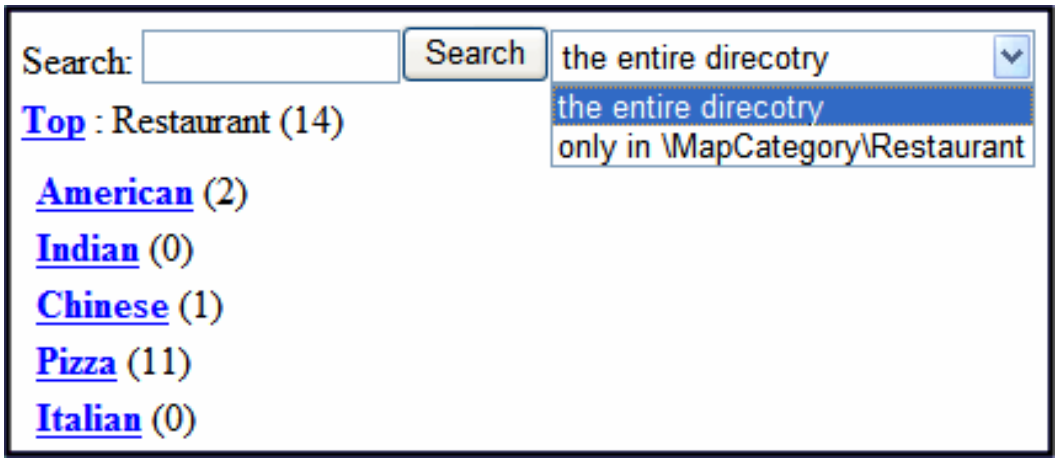
  This property, in conjunction with the **MaxNumber** property, lets site visitors view an unlimited number of items while controlling the amount of screen space. The **MaxNumber** property limits the number of items displayed. If you set this property to **True**, and the number of items exceeds **MaxNumber**, navigation aids appear below the last item, allowing the visitor to go to the next screen. See example below.

So, for example, if a taxonomy has 9 items and `MaxResults` is set to 3, the screen displays only the first 3 items. When the site visitor clicks **[Next]**, he sees items 4, 5 and 6, and so on.

IMPORTANT: If the `EnablePaging` property is set to True, the `CacheInterval` property is disabled.

- **EnableSearch** (Boolean)

  Set to **True** if you want a search box (shown below) to appear above this taxonomy display. A site visitor can use the search to find content within a taxonomy/category that includes terms of interest. To suppress the search box, set to **False**.



- **GetAnalyticsData** (Boolean)

  Set this property to **True** if you want the following information for each content in the list. Returns **Content View Count**, **Content Rating**, **Content Rating Average**. Create your own XSLT styles to display this data.

  IMPORTANT: This property provides reliable data only when the Business Analytics Feature is on. See Running Ektron Business Analytics.

- **Hide** (Boolean)

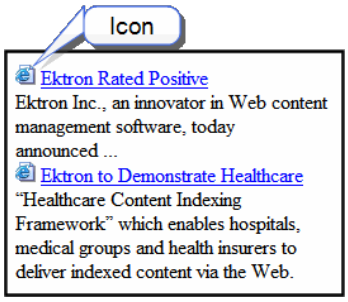
  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **IncludeIcons** (Boolean)

  Choose whether to display icons next to the taxonomy's links.

  This property only works when ecmSummary or ecmTeaser are used in the `DisplayXslt` property. When the `[$ImageIcon]` variable is used in an EkML file and that file is assigned to the `MarkupLanguage` property, this property acts as True.



  - **True**. Show icons
  - **False**. Hide icons
- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **LinkTarget** (ItemLinkTargets)

  Determines the type of window that appears when you click a link in the server control.

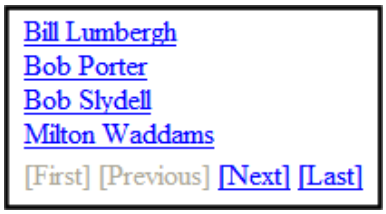
  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **MarkupLanguage** (String)

  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.See also: *Ektron Markup Language* on page 2633

  See also: *taxonomy.ekml* on page 2641

  NOTE: If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored. If the EkML file contains the `[$ImageIcon]` variable, the `IncludeIcons` property acts as True.
- **MaxResults** (Integer)

  Enter the maximum number of items to appear in the initial display of this server control. If you set this value to zero (0), the maximum is 50. To let site visitors view more than the maximum but limit the amount of space being occupied, enter the maximum number of results per page here. Then, set the `EnablePaging` property to **True**. If you do and more than the number of `MaxResults` are available, navigation aids appear below the last item to help the site visitor view additional items. See following example below.

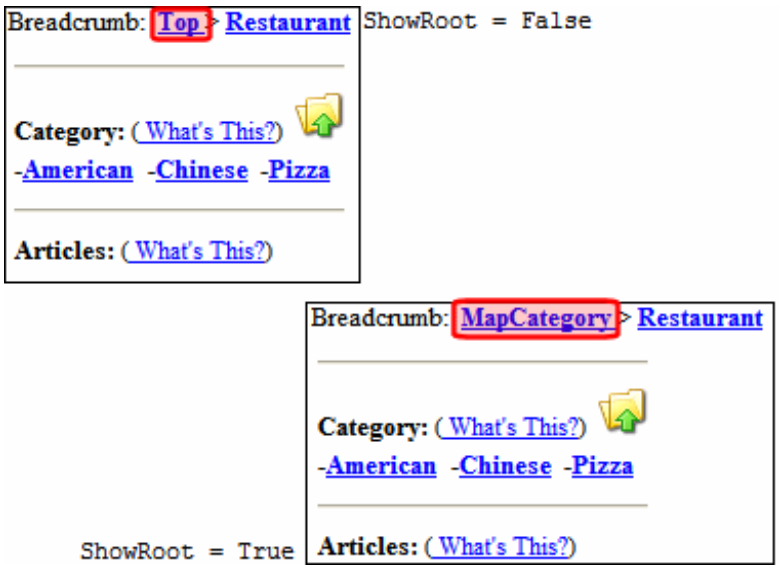


- **OrderItemsBy**
  Specify the sort order of results. Choices are:
  - **taxonomy_item_display_order**. The order of taxonomy items as set in the Workarea.
  - **content_title**. The content is listed in alphabetical order by title.
  - **date_created**. Content is listed in the order by which it was created.
  - **last_edit_date**. Content is listed in order, by its last edit date.
  - **content_rating_average**. Business Analytics Content Rating
  - **content_view_count**. Business Analytics Content Views

  Use the the `SortDirection` propertyYou can to specify the direction of the items with the `SortDirection` property.
- **PreserveUrlParameters** (Boolean)
  Set to `true` if this control needs to preserve the URL parameters on the page. For example, if a server control displays a list of content items, the URL parameter passes to the control the template names and `id` values that the template expects. Set to `false` if the server control does not need to pass the URL parameters to the server control. The default value is `false`.
- **Recursive** (Boolean)
  Set to True to include child folders of the parent folder.
- **ShowAllChecked** (Boolean)
  When set to `true`, a check box appears in the **Show All** check box.

  Also, set this property to `true` if you want categories to which no items are assigned to appear. If set to `false`, only categories to which items are assigned show up in the display.
- **ShowCount** (Boolean)
  Indicates if the number of taxonomy items appears next to each category when displayed in the website. (See example below.) Default value is false.

- **ShowRoot** (Boolean)
  - **False. Top** represents the first node of the taxonomy path.
  - **True**. The *name* of the taxonomy path's first node appears instead of **Top**.



- **SortDirection** (String)

  Select the direction of the `ItemSortOrder` property. Choose Ascending or Descending.

- **TaxonomyCols** (Integer)

  Enter the number of columns in which this taxonomy/category will appear on the page.

- **TaxonomyDepth** (Integer)

  Enter the number of taxonomy levels to retrieve below each taxonomy/category if you are accessing a taxonomy's XML using code-behind. For example, if the taxonomy is **Clients > Industries > Education > Colleges**, and you set **Taxonomy Depth** to **2**, only Clients > Industries are available in code-behind.

  Exception: If you set this value to 1, the control retrieves taxonomy levels to a depth of 2. For example, if the taxonomy is **Businesses > Restaurants > Fast Food > Pizza**, and you set **Taxonomy Depth** to **1**, Business and Restaurants are available in code-behind.

This setting has no effect on the display generated by the Directory server control, which always displays one level below the current. To retrieve all categories for a taxonomy recursively, enter -1. The default value is 1.

IMPORTANT: For a live site, leave this value at **1**. Increasing this value can slow down your live Web server. However, for testing on a staging server, you can increase the depth.

- **TaxonomyId** (Long)
  Enter the ID number of the taxonomy or category to appear in this server control. If you don't know the number, click the button and navigate to the taxonomy or category.
- **TaxonomyItemCols** (Integer)
  Enter the number of columns in which this taxonomy/category items (articles) will appear on the page.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

eCommerce server controls

8.50 and higher

Ektron provides a set of eCommerce server controls that let you set up an online marketplace where customers (site visitors) can purchase merchandise, services or content. Customers interact with these server controls on your site.

- *Cart* on page 2359. View products in a shopping cart, create new carts, and apply discounts.
- *Checkout* on page 2367. Check out.
- *CurrencySelect* on page 2381. Select a currency.
- *MyAccount* on page 2384. View and change their account information.
- *OrderList* on page 2390. View a list of processed orders.
- *Product* on page 2396. View a product's details and add it to their shopping cart.
- *ProductList* on page 2405. Search for products or view a list of products.
- *Recommendation* on page 2413. Get recommendations on other products.

Most eCommerce server controls let customers move from one control to another via template properties that let you define the location of another server control. For example, if you define the path to the Cart server control in the Product server control's TemplateCart property, customers are directed to the Cart control when they click the Product control's Add to Cart button.

Ektron's server controls let you insert standard methods and properties within the Visual Studio environment. This means that you can see the effect of your changes in real time; you don't have to modify a page and then compile to see results.

You can insert server controls using drag and drop or programmatically. You can also use databinding to retrieve and display data from Ektron.

Ektron provides a complete set of eCommerce server controls that let you set up an online marketplace where customers can purchase merchandise, services or content.

The following list shows typical actions a customer might perform when visiting your eCommerce site and how you can facilitate these actions.

- Viewing account information. Create a link in the master page that leads to the template containing the *MyAccount* on page 2384 server control. This link should be available from any page.
- Viewing previous and current orders. Create a link in the master page that leads to the template containing the *OrderList* on page 2390 server control. This link should be available from any page.
- Changing the currency type. Add the *CurrencySelect* on page 2381 server control to a master page. This server control should be available from any page.
- Searching for products. Either create a link to the template containing the *ProductSearch* on page 2085 server controls. Or, on the master page, add a text box that passes the text to a hidden ProductSearch server controls and post the results to a separate template. This link or text box should be available from any page.
- Viewing a list of products. Create a template that contains the *ProductList* on page 2405 server control. This template might be the first page customers see when they view the eCommerce portion of your site. Whenever you want to display a list of products based on a catalog, taxonomy or a list content IDs. For example, you could create a list of products in a panel on the right side of your website.
- Viewing a product's details. Create a template that contains the *Product* on page 2396 server control. Use this whenever a customer needs to view details of a product. Set any of the following server control's TemplateProduct property to the path of this control. This makes a product's title being displayed by one of these controls a clickable link that takes the customer to this control.
  - ProductSearch
  - ProductList
  - OrderList
  - Cart
  - Recommendation
- Viewing additional products associated with another product. Add the *Recommendation* on page 2413 server control to a template that contains a Product server control and set the Recommendation control's DynamicProductParameter property to the QueryString parameter that's used

to pass a products ID. When ever you want to present a customer with cross-sell and upsell opportunities associated with a given product.

- Adding a product to their shopping cart. Set the TemplateCart property in the following server controls to the path of the *Cart* on the facing page server control. Setting this property is one of the requirements for the Add to Cart button or link to appear in these controls.
  - ProductSearch
  - ProductList
  - Product
  - Recommendation

  NOTE: In addition to setting the above property, a product must be in-stock, not archived and buyable. Otherwise, the Add to Cart button or link does not show for a product.

  - Use this when you want to let customers add products to their shopping cart.
  - Viewing products in their shopping cart. When a customer adds a product to their cart, they are sent to the Cart server control. You should also create a link in the master page that leads to a template containing this control. The link should be available from any page.
- Checking out and paying for their selected product. Add the *Checkout* on page 2367 server control to a template and add that template's path to the Cart server control's TemplateCheckout property. When you do, the Checkout button in the Cart server control becomes active. You must set this up for customers to checkout.

Customizing eCommerce server controls with event hooks

The eCommerce server controls contain event hooks so you can customize or manipulate them. To use these event hooks, create a handler for each needed event in the code-behind of the ASPX page that contains an eCommerce server control. Then, in that page's `OnInit` method, hook each event.

- **PhaseChange**. Used only in the Checkout server control and called when the current phase is about to change. Supervise phase changes and allow or prevent changing as needed. It also lets you jump to a specified phase. For example, you could jump to a passed phase, prevent the phase from changing from the current one under certain conditions, or jump to a phase that is not necessarily the next one in line.
- **PreProcessXml**. Called immediately before the control's XML is transformed. Inspect or manipulate the server control's XML before being processed. This is the only time the XmlDoc property should be used to access the controls XML.
- **PostProcessXml**. Called immediately after the control's XML is transformed. Work with processed transformed XML output inside the Text property before

sending it to the browser. This allows simple string manipulation of the server control's markup.

- **FilterAjaxCallback**. Called when an Ajax-Callback event occurs. Inspect or modify callback parameters. For example, you could use this event to receive custom values entered in a page.
- **PostLogin**. Used only in the Checkout server control and called immediately after a (possibly new) user logs in with this control. Calls custom code immediately after a returning user logs in with an email and password. Or, immediately after a new customer fills out the billing info screen. For example, you could add code that lets new customers sign up for Web Alerts after they add their information to the billing screen.

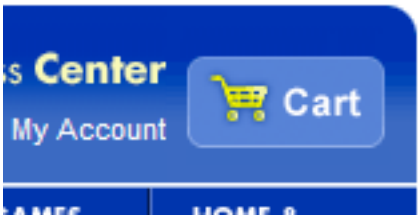
Cart

8.50 and higher

The Cart server control lets a customer (site visitor) select products to purchase; an online "shopping basket." Products are added to the cart when the customer selects them.

A customer typically reaches the Cart server control from a product description or list page. A product description page contains a *Product* on page 2396 server control. Product list pages contain server controls that create a list of products, such as *ProductList* on page 2405, *ProductSearch* on page 2085 or *Recommendation* on page 2413 server controls. These controls contain a button or link that allows a customer to add the product to the cart.

You could also create a link to the cart from a master page or menu that lets customers view their cart. This link lets a customer navigate directly to the cart when they arrive at your site.



Inserting the Cart server control onto a page

PREREQUISITE

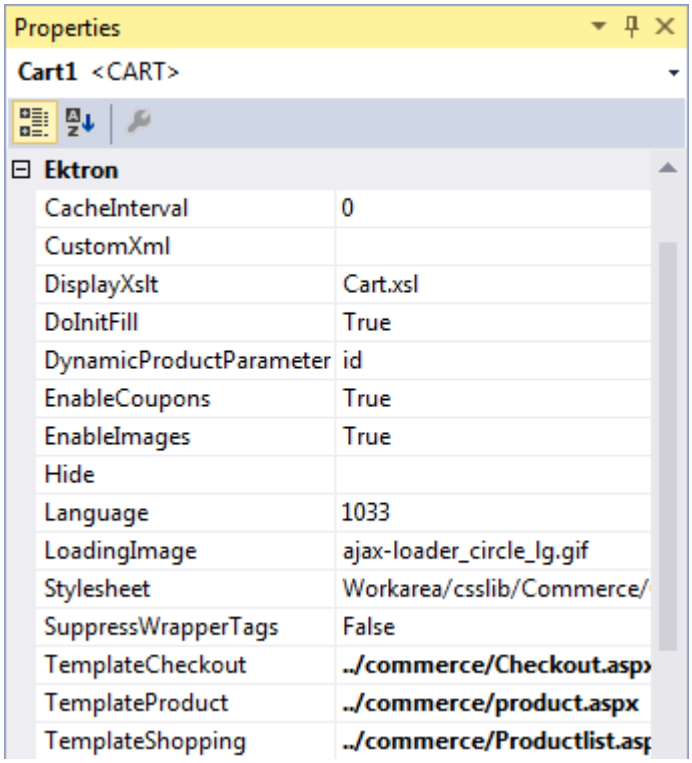
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Cart** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Cart ID="Cart1" runat="server" />
```

4. Click on `Cart` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



Cart properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **CustomXml** (Code-behind Only) (String)

  Lets you inject custom XML into the generated XML before being processed by the XSLT. Enter a string of XML that you want to make available to the XSLT as follows:

  ```
  <root><customXml>custom-xml-inserted-here</customXml></root>
  ```

  See also: *Displaying custom XML in Ektron's server controls* on page 2162.

- **DisplayXslt** (String). If desired, enter a relative or absolute path to an Xslt that determines the display of the page. By default, the control uses *siteroot*\Workarea\Xslt\Commerce\Cart.xsl.

  WARNING! Files stored in the siteroot\Workarea folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the siteroot\workarea folder then edit it. If there is no default file, create the file outside the siteroot\workarea folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean). Controls when Fill occurs.

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicProductParameter** (String). The QueryString parameter name which is used to pass the product ID to the product details page. For example, if your QueryString parameter for products is ID, enter that in this property. Then, when a customer clicks a product's title, this parameter is passed with the product's ID to the product details page.
- **EnableCoupons** (Boolean). Set to true to allow a customer to enter coupon codes for discounts. When set to false, the **Apply Coupon** button is hidden. Default value is True if this property is not set. See also: Using Coupons.
- **Hide** (Boolean). Used to hide the control in design time and run time.
  - **True**. Hide the control
  - **False**. Show the control
- **Language** (Integer).

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LoadingImage** (String). The image to display while the cart is loading. The default image is *siteroot*\Workarea\images\application\ajaxloader_circle_lg.gif.

  WARNING! Files stored in the siteroot\Workarea folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the siteroot\workarea folder then edit it. If there is no default file, create the file outside the siteroot\workarea folder. Next, in this property, enter the path to that file relative to the site root folder.

- **Stylesheet** (String). Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the siteroot\workarea folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder. The default style sheet is ***siteroot*\Workarea\csslib\Commerce**cart.css.
- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TemplateCheckout** (String). The template that contains the Checkout server control. A customer is directed to this page when he clicks **Checkout**. If the template file is located in the same folder as the Web form containing this server control, just enter its name. The path can be relative or absolute. See also: *Checking out* on page 2367.
- **TemplateProduct** (String). The template that contains the Product server control. A customer is directed to this page when a product in the cart is clicked. This page provides details about the product. See also: *Displaying a product* on page 2400.

  If this property is blank, the server control uses the Product's QuickLink information.

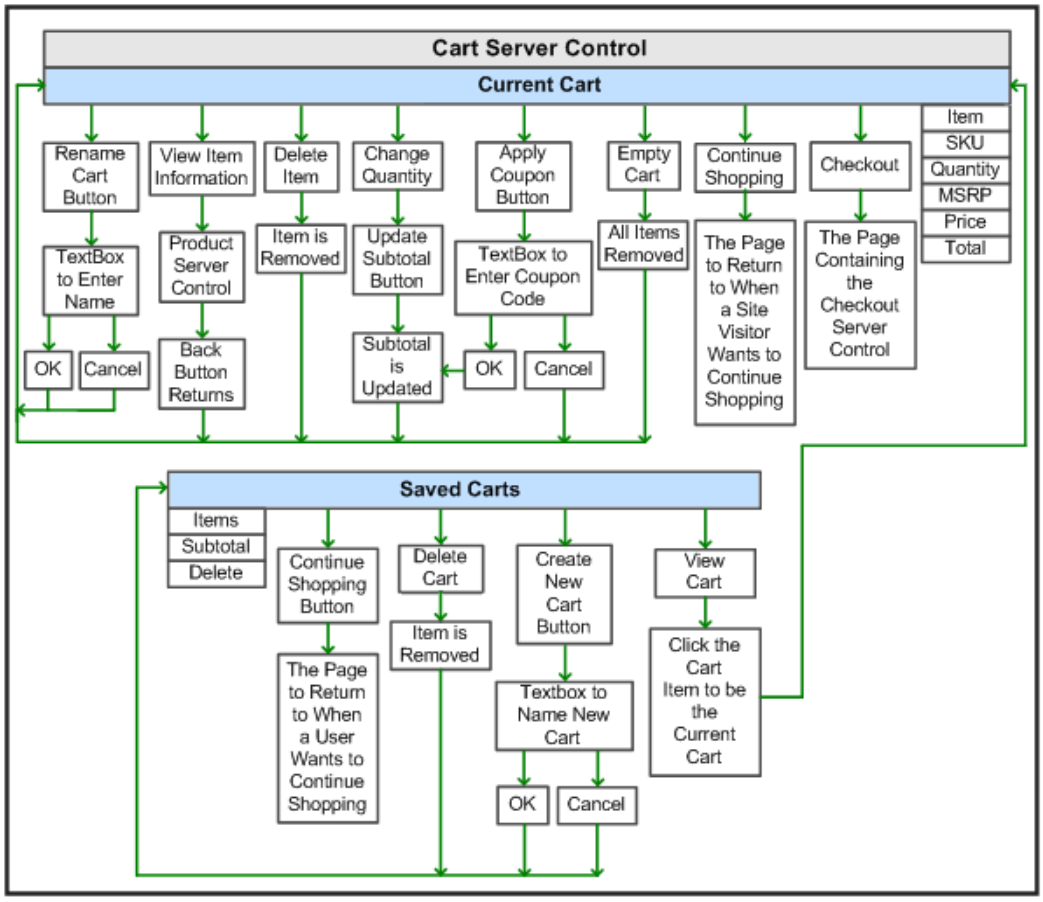
  If the template file is located in the same folder as the Web form containing this server control, just enter its name. The path can be relative or absolute.

- **TemplateShopping** (String). The URL to navigate to when the customer clicks **Continue Shopping**. This URL could direct the customer to a template containing a ProductSearch or ProductList server control, which lets the person select additional products. See also: *Continuing to shop* on page 2366.

Items are added to a cart when a customer clicks **Add to Cart** in the following server controls.

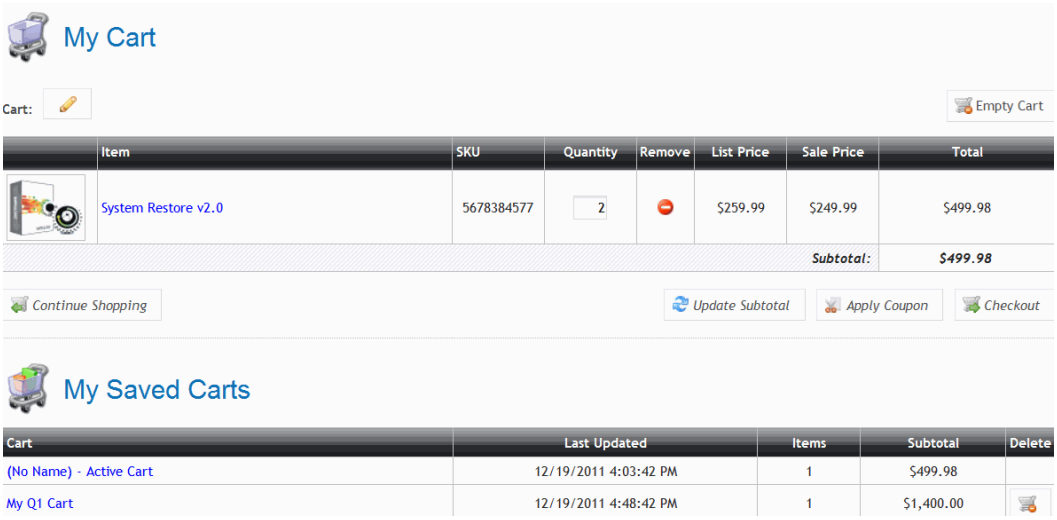
- *Product* on page 2396
- *ProductList* on page 2405
- *ProductSearch* on page 2085 templated server controls
- *Recommendation* on page 2413

When **Add to Cart** is clicked, the item is appended to the current cart's list of items. If the customer does not have a cart, one is created. The following image depicts the flow of the Cart server control.



Cart server control areas

The Cart server control consists of 2 major areas. The top part (**My Cart**) represents the cart with which the customer is currently working. The bottom (**My Saved Carts**) represents a visitor's saved carts. Saved cart information appears only when a customer is logged in. See also: *Creating a cart* on page 2365.



The **My Cart** area displays the Item, SKU, Quantity, List Price, Sale Price, Subtotal and Total. From this cart, a customer can remove items, update the quantity and subtotal information, apply coupons and check out. The customer can also choose to continue shopping or empty the cart.

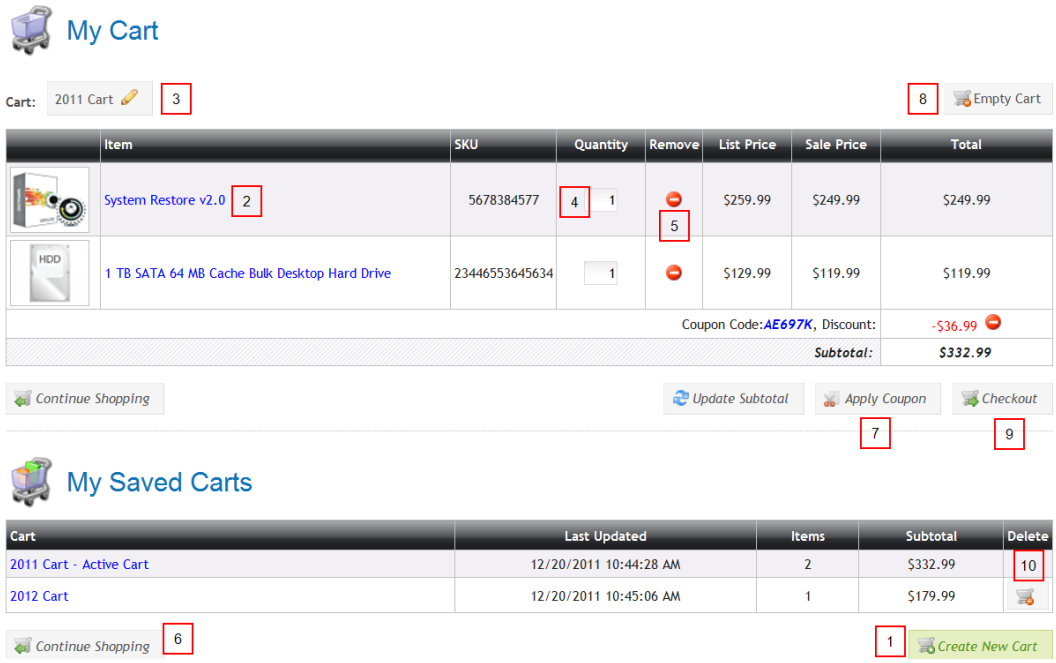
The **My Saved Cart** area contains a list of carts the customer has saved and are awaiting checkout. This allows the customer to select products and save them for future purchase. This area contains:

- the name of each cart
- the last time it was updated
- how many items in each cart
- the subtotal of each cart
- options to
  - delete a saved cart
  - create a new cart
  - continue shopping

Clicking a saved cart makes it the current cart, and displays its products in the **Your Cart** area. A customer can then proceed to check out.

Working with a Cart

This section explains how a logged-in customer would use the Cart server control. The following figure helps you locate key features of the Cart.



1. Create New Cart button.
2. Item name link.
3. Cart name edit button.
4. Item quantity field.
5. Delete item button.
6. Continue Shopping button.
7. Apply coupon button.
8. Empty Cart button.
9. Checkout button.
10. Delete Cart button.

Creating a cart

A customer automatically creates a new cart the first time a product is added to it. If needed, a customer can create multiple carts while using the Cart server control. This allows for the grouping of products being purchased. For example:

- customers create carts based on concepts, such as, Son's Birthday or Item Type.
- customers use separate payment types. For example, a customer places some products on one credit card and the rest on another. By having 2 carts, a customer can proceed with 2 separate checkouts.
- customer needs to prioritize purchases. For example, a customer wants to purchase certain products, but cannot afford them right now. The customer can add these products to a separate cart, then continue to purchase affordable products now. This helps customers remember purchases they want to make, driving additional sales for your site.

To create a new cart:

1. Click **Create New Cart** in the **Saved Cart** area. (See Callout 1.) The button is replaced with a text box.
2. Enter the name of the new cart in the text box.
3. Click **OK**.

When the customer clicks **OK**, the cart is added to the list of Saved Carts and becomes the active cart. A customer can then click **Continue Shopping** to select products to add to the new cart.

Displaying an item's information

In the Cart server control, a product's title is a link. When clicked, it takes you to the product's detail page. (See Callout 2.) To navigate back to the cart, click your browser's Back button.

By default, the server control uses the product's QuickLink information to provide a path to the product. You can override this functionality by adding a new path to the TemplateProduct property.

Assigning or changing the name of the cart

Assigning a name to a cart makes it easier for customers to identify a cart in their saved cart list. A customer can assign or change the name of a cart by clicking **Edit** () next to the cart's name. (See Callout 3.)

When clicked, the button and name are replaced with a text box that allows the customer to enter a new name for the cart.

Changing an item's quantity

To change a product's quantity, enter the new amount in the **Quantity** column's textbox, located in the same row as the product you want to change. (See Callout 4.) Next, click **Update Subtotal**. This recalculates the price of the product and updates the subtotal. The image below shows the Atrium Table quantity change to 3.

Removing an item from the cart

A customer can remove a product from the cart by clicking **Remove From Cart** () in the **Quantity** column. (See Callout 5.) When the customer clicks the button, the product is removed from the cart, and the subtotal is updated.

Continuing to shop

A customer can continue to shop by clicking **Continue Shopping**. (See Callout 6.) This redirects the visitor to a template defined in the cart server control's `TemplateShopping` property. For example, you might send the customer to a page containing a *ProductList* on page 2405 server control or *ProductSearch* on page 2085 templated server controls.

As a developer, you need to add the path to this page to the `TemplateShopping` property. If the page is in the same folder as the page that contains the Cart server control, just enter the page's name.

Applying coupons

If coupons are defined in the Workarea and the `EnableCoupons` property is set to True, a customer can enter coupons to discount the purchase. How coupons affect the purchase is defined in the **Workarea > Settings > Commerce > Catalog > Coupons**. See also: Using Coupons.

To apply a coupon, a customer clicks **Apply Coupon**. (See Callout 7.)

The button area changes to text box where the customer can enter the **Coupon Code**. The customer then enters a code and clicks **OK**. Next, the discount appears above the **Subtotal**, and the customer can continue to shop, check out, view other carts, or delete the coupon by clicking **Remove Coupon**.

Emptying the current cart

To remove all products from the current cart, a customer clicks **Empty Cart**. (See Callout 8.) When clicked, a dialog box appears and the customer can click **OK** or **Cancel**.

Checking out

When a customer is satisfied that the cart contains all products they want, the quantity of each product is correct and all coupons have been applied, the customer can click **Checkout**. (See Callout 9.) At that point, the customer is redirected to the page that hosts the Checkout server control. The path to the page is defined in the `TemplateCheckout` property. If the page containing the Checkout server control is in the same folder as the Cart server control, just enter the page name.

If a cart includes an item that is out of stock, the **Checkout** button is disabled (hidden).

Deleting a saved cart

A customer can delete a saved cart by clicking **Delete Cart**. (See Callout 10.)

Checkout

8.50 and higher

Customers typically arrive at this server control by clicking **Checkout** on the Cart server control. They also might reach this server control from a Checkout link you create on your site. The Checkout server control lets a customer navigate through the checkout process. This process includes:

- adding billing and shipping information
- selecting a shipping method
- reviewing the order
- submitting the order and credit card information
- notification that checkout process is complete

IMPORTANT: When using the Checkout server control, your website should have an SSL certificate installed. Set its `IsSSLRequired` property to **True**. This protects your

customers' payment information during transmission. Do not use a self-signed SSL certificate; obtain one from a trusted certificate authority.

The control provides a navigational aid that indicates the customer's step in the checkout process. At any point, the customer can move forward to the next step or back to a previous one. When appropriate, some steps do not appear. For example, if merchandise is not tangible, the shipping screen does not appear.

Inserting the Checkout server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

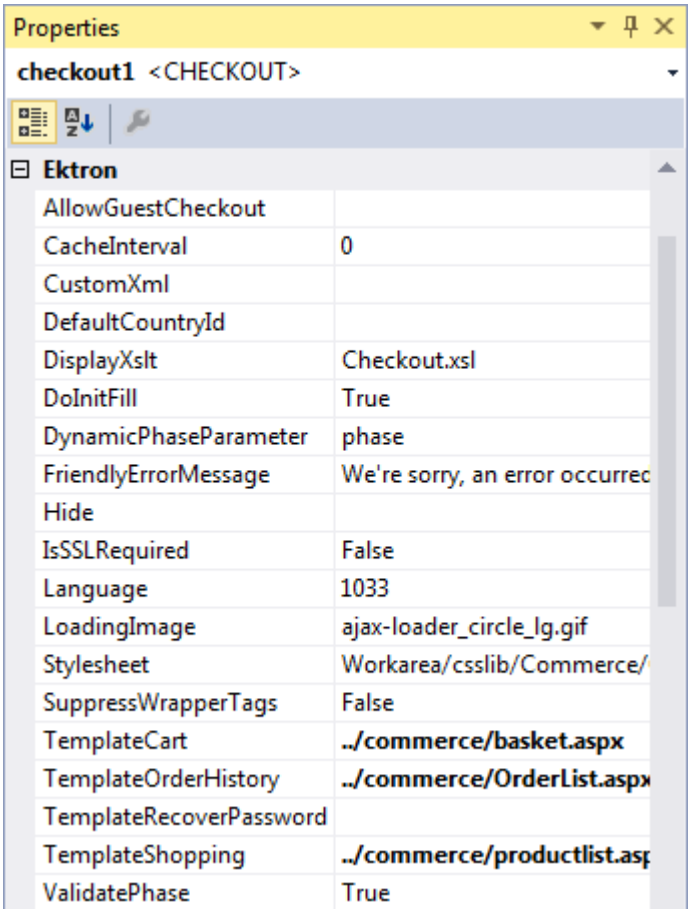
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Checkout** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:Checkout ID="Checkout1" runat="server" />
   ```

4. Click on `Checkout` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



Checkout properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AllowGuestCheckout** (Boolean)
  Set to true if you wish to allow the Guest Checkout feature.
- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **CurrentPhase** (Code-behind Only) (ControlPhase)
  Reports or selects the current page (or phase) of the checkout control. This lets you customize the flow of the control. For example, if you only offer one shipping option, skip the ShippingInfoEntry phase.
  - **Login**. Allows existing users to log in. *Logging in or setting up an account* on page 2373.
  - **BillingInfo**. Displays a customer's billing information. See also: Billing Information.

- **BillingInfoEntry**. Allows entry of a customer's billing information. See also: Billing Information.
- **ShippingInfo**. Allows entry of a customer's shipping information.
- **ShippingInfoEntry**. Allows entry of a customer's shipping information. See also: *Shipping information* on page 2377.
- **ShippingMethodSelect**. Allows selection of the shipping method. See also: *Shipping method* on page 2378.
- **ReviewOrder**. Displays the current order's details. See also: *Review order* on page 2378.
- **SubmitOrder**. Lets customers add payment information. See also: *Submit order* on page 2379.
- **Complete**. Displays completion message. See also: *Order complete* on page 2380.
- **Error_EmptyBasket**. Used when a cart has no items and the customer somehow enters the control.
- **Error_UnhandledException**. Occurs when there's an error that the control cannot resolve.

- **CustomXml** (Code-behind Only) (String)

  Lets you inject custom XML into the generated XML before being processed by the XSLT. Enter a string of XML that you want to make available to the XSLT as follows:

  ```
  <root><customXml>custom-xml-inserted-here</customXml></root>
  ```

  See also: *Displaying custom XML in Ektron's server controls* on page 2162.

- **DefaultCountryId** (Integer)

  The ID of the default country that appears in the Billing Address and Shipping Address. Set this to the country from which the majority of your customers make purchases. If a customer is from a different country, he can change it when editing the Billing or Shipping Address.

  To find a country's numeric ID, go to **Workarea > Settings > Commerce > Configuration > Countries**. The Numeric ID is in the left column of that screen.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page. By default, the control uses `Checkout.xsl`. This file is located in *`siteroot`*`\Workarea\Xslt\Commerce\Checkout\Standard`.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicPhaseParameter** (String)

  The name of the parameter on the QueryString that identifies desired phase ID.

- **FriendlyErrorMessage** (String)

  The message displayed when an unhandled error occurs. Details are sent to the event log.

  The default message is "*We're sorry, an error occurred while processing your request. Please try again later...*"

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **IsSSLRequired** (Boolean)

  When set to True, switches to an SSL encrypted URL. For the SSL encryption to work, you must have an SSL certificate installed for your site. See also: Updating web.config to use SSL. Information on SSL can be found at the following websites.

  - Secure Sockets Layer (SSL) (Tech-FAQ)
  - Introduction to SSL (Symantec)

  NOTE: Episerver is not associated with TechFAQ or VeriSign. However, both sites offer good explanations of Secure Sockets Layer.

  Installing and using an SSL is one of the most important things you can do to protect your customer's credit card data.

- **Language** (Integer)

  Set a language for viewing the checkout control. This property shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LoadingImage** (String)

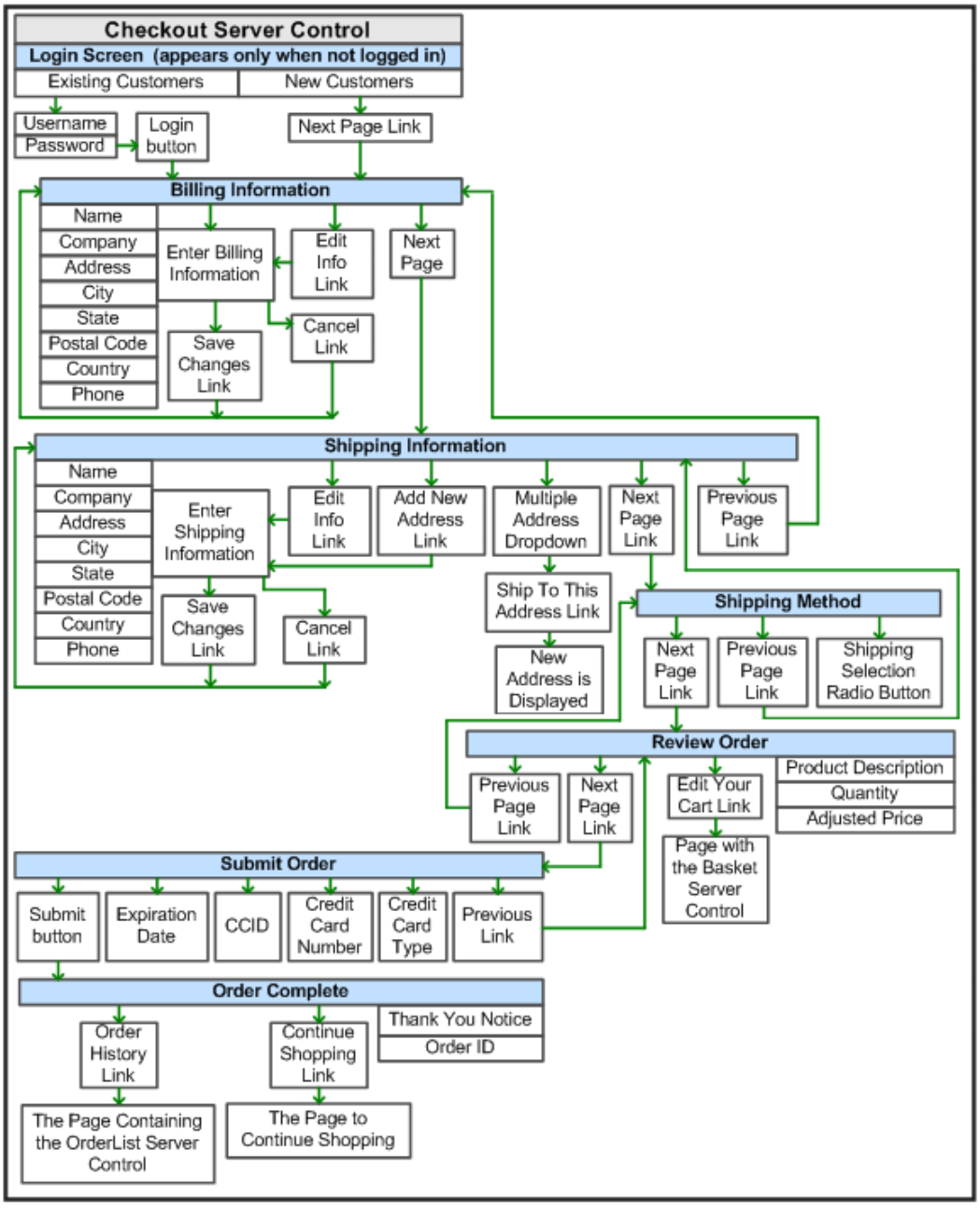
  The image to display while the control fetches data. The default is `siteroot\Workarea\images\application\ajax-loader_circle_lg.gif`.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.
- **SuppressWrapperTag** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TemplateCart** (String)
  The URL path of the template that contains the *Cart* on page 2359 server control. The path can be relative or absolute.

  When a path is entered, a link appears in the Review Order part of the process that allows a user to navigate to the template containing the Cart server control.
- **TemplateOrderHistory** (String)
  The URL path of the template that contains the *OrderList* on page 2390 server control. The path can be relative or absolute. When a path is entered, a link appears at the end of checkout process that allows a user to navigate to the template containing the OrderList server control.
- **TemplateRecoverPassword** (String)
  The URL path of the template that helps a customer recover a password. The path can be relative or absolute. When a path is entered and a customer has not logged in, a link appears at the beginning of checkout process that lets the customer navigate to a template containing information to recover the password. The Membership server control contains an option for recovering passwords. See also: Logging In or Setting Up an Account and *Membership* on page 2482 server control.
- **TemplateShopping** (String)
  The URL path of the template that allows the customer to continue shopping. If a path is entered, a link appears at the end of checkout that allows a user to navigate to a template that lets the person continue shopping; for example, the template containing a *ProductList* on page 2405 server control or *ProductSearch* on page 2085 server controls.

The following chart represents the options and processes of the Checkout server control.



Logging in or setting up an account

The login portion of the Checkout server control accommodates these types of customers.

- **Existing customers**

  If an existing customer is not logged in when the customer reaches this server control, it prompts the customer to log in. Optionally, this screen could have a link that lets an existing user to recover a password. To enable this link, add a template path to the control's `TemplateRecoverPassword` property.

  After a customer who previously purchased from this site logs in, billing and shipping information is retrieved and appear by default. The visitor can review

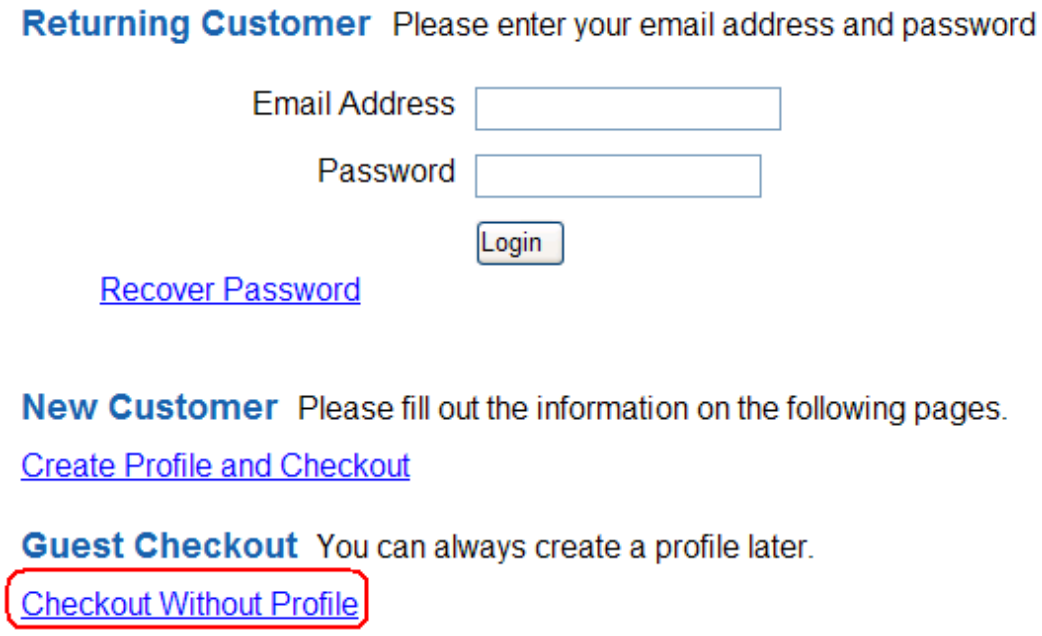
and edit this information. However, Ektron does not store credit card numbers, so that information must be entered each time.

- **First-Time Customer**. Customers who have not logged in before but who wish to create an account

  If a customer has not purchased from your site before and wants to set up an account, the customer clicks **Create Profile and Checkout**. On the new screen, the customer enters billing information including a password. After completing that screen, a membership user account is created, and the customer is logged in. The new user can then proceed through the checkout. The next time this visitor logs in, the customer is treated as an existing customer.

- **Guest Account Registration. C**ustomers who wish to make one-time only purchase and not set up an account.

  The Checkout server control provides a guest checkout feature (circled below) for customers who do not want to set up an account on your site. To enable this feature, set the `AllowGuestcheckout` property of the Checkout server control to `true`.



If a user clicks **Checkout Without Profile**, the customer proceeds to a version of the billing screen that does not ask for a password. The rest of the checkout is the same.

Although the customer can proceed through the checkout like other customers, if the customer wants to checkout in the future, the customer must re-enter billing and shipping information. Customers who use the guest checkout feature can view their orders using the Order List server control if its `GuestOrderView` property is set to `true`.

Checkout screens

The Checkout server control has several sections, each representing a portion of the checkout process.

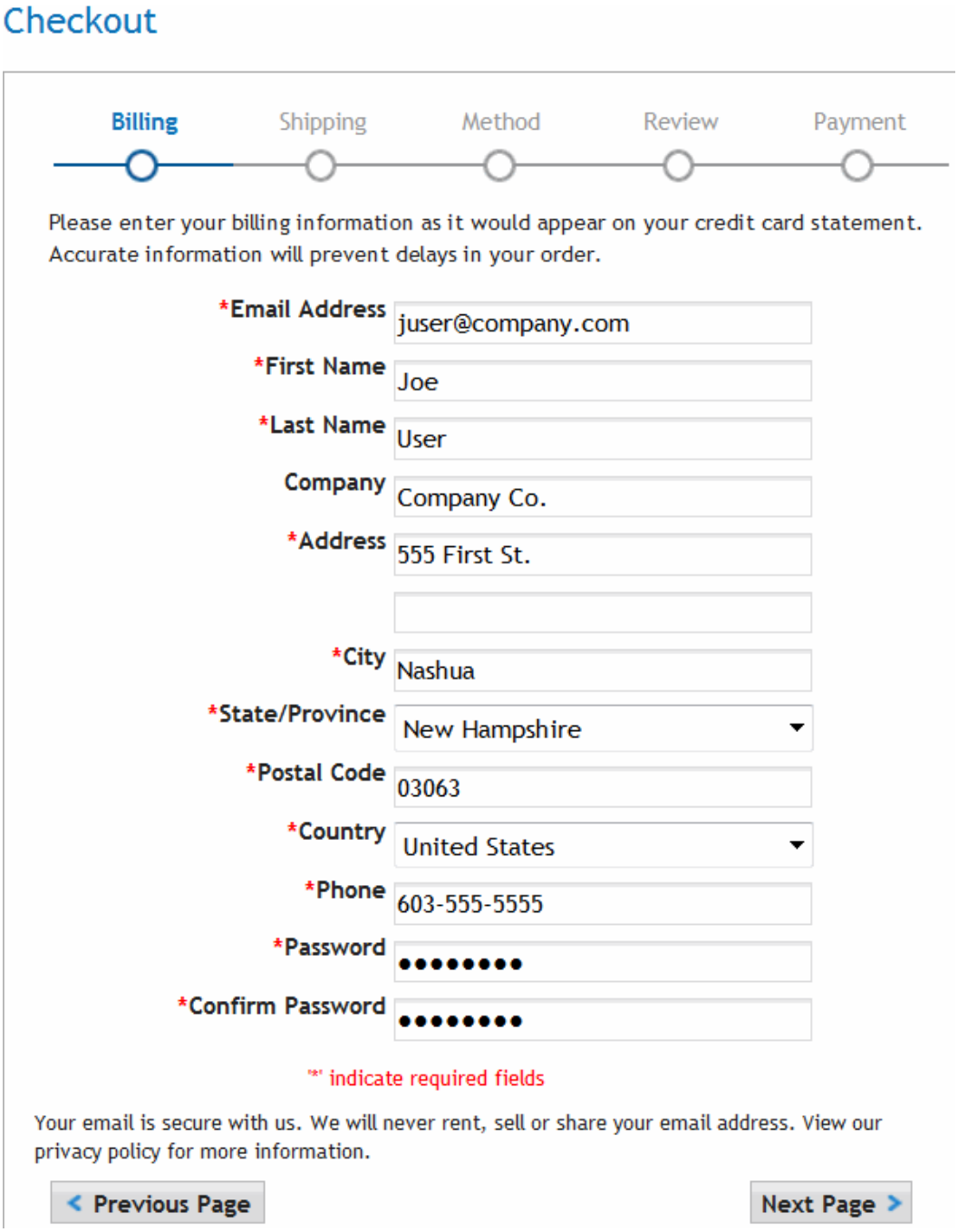
Billing information

Prompts customers to enter or update billing information, including:

- Name
- Company (Optional)
- Street Address, City, State, Postal Code, Country

  NOTE: If a customer chooses a country other than the United States, no field validation is applied to the postal code by default. To apply country-specific postal code validation, see *Applying field validation for non-U.S. postal codes* on the next page.

- Phone
- email address (appears when a visitor is creating a new account or doing a guest checkout)
- Password (appears when a visitor is creating a new account or doing a guest checkout)



Applying field validation for non-U.S. postal codes

1. Search the XSLTs for `EktronCheckout_PostalField`. The standard US Postal RegEx is applied to all input fields of this class.
2. Remove that class.

   WARNING! Be careful to remove only this class name; otherwise, the CSS styling may not be applied properly and the layout will be altered.

3. Add an attribute to each postal-code input field in the XSLTs with the following name: `data-ektron-validation-regex`. Then, add the desired RegEx as the value of this attribute. The RegEx expression will validate the user's input.

For the RegEx expression, you should be able to find one for the target country via a Web search. Alternatively, you can start here: http://www.regular-expressions.info/.

Canadian postal code solution

The following example shows how to apply field validation for Canada.

Attribute for U.S. postal code regex:

```
<xsl:attribute name="data-ektron-validation-regex">/^\d{5}(-\d{4})?$/
</xsl:attribute>
```

Attribute for Canadian postal code regex:

```
<xsl:attribute name="data-ektron-validation-regex">/^([ABCEGHJKLMNPRSTVXY]
  \d[ABCEGHJKLMNPRSTVWXYZ])\ {0,1}(\d[ABCEGHJKLMNPRSTVWXYZ]\d)$/
</xsl:attribute>
```

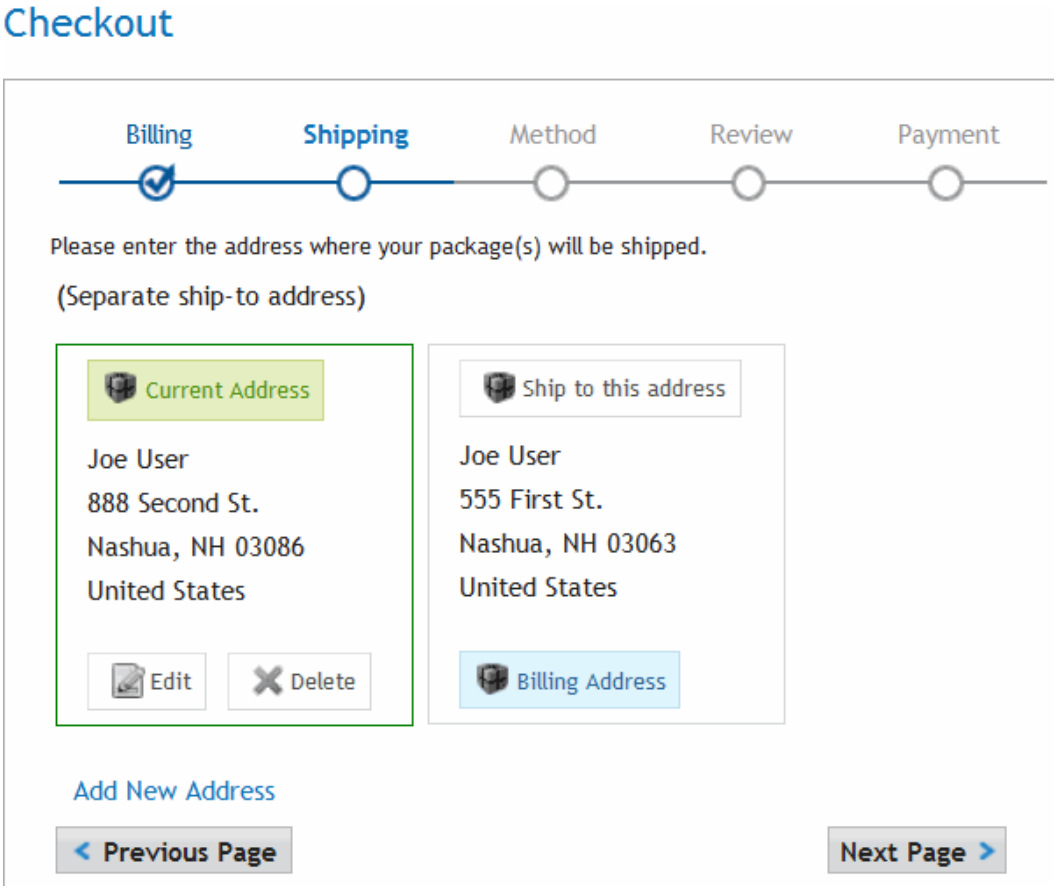
The following sample shows the entire postal code input block taken from billing info XSLT. Three checkout-related files that need these changes (`Checkout_BillingInfo.xsl`, `Checkout_BillingInfoEntry.xsl` and `Checkout_ShippingInfo.xsl`) are updated in the sample below for Canadian only postal codes.

```
<input class="EktronCheckout_PostalCode EktronCheckout_Row_RightContents" >
  <xsl:attribute name="name">EktronCheckout_BillingInfo_PostalCode_
    <xsl:value-of select="$ControlId"/></xsl:attribute>
  <xsl:attribute name="id">EktronCheckout_BillingInfo_PostalCode_
    <xsl:value-of select="$ControlId"/></xsl:attribute>
  <xsl:attribute name="value"><xsl:value-of select="/root/billingInfo/postalCode"/>
    </xsl:attribute>
  <xsl:attribute name="data-ektron-validation-regex">/^([ABCEGHJKLMNPRSTVXY]
    \d[ABCEGHJKLMNPRSTVWXYZ])\ {0,1}(\d[ABCEGHJKLMNPRSTVWXYZ]\d)$/</xsl:attribute>
</input>
```

Shipping information

Add or edit shipping information. By default, it uses the billing address as the shipping address. From this screen, customers can:

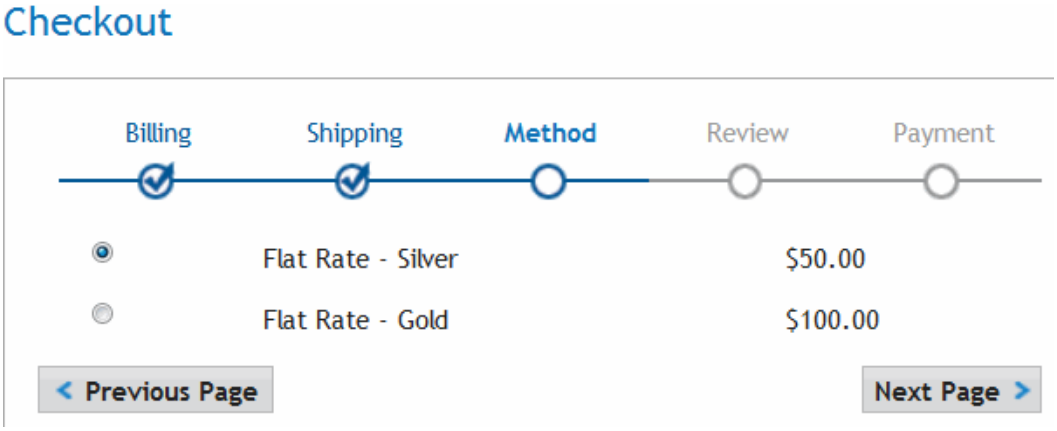
- add new addresses
- edit or delete existing addresses
- if more than one address is entered, pick address for current order
- proceed to the next step or back to the previous page



When a logged-in customer adds a new address, it is stored with the account information in Ektron.

Shipping method

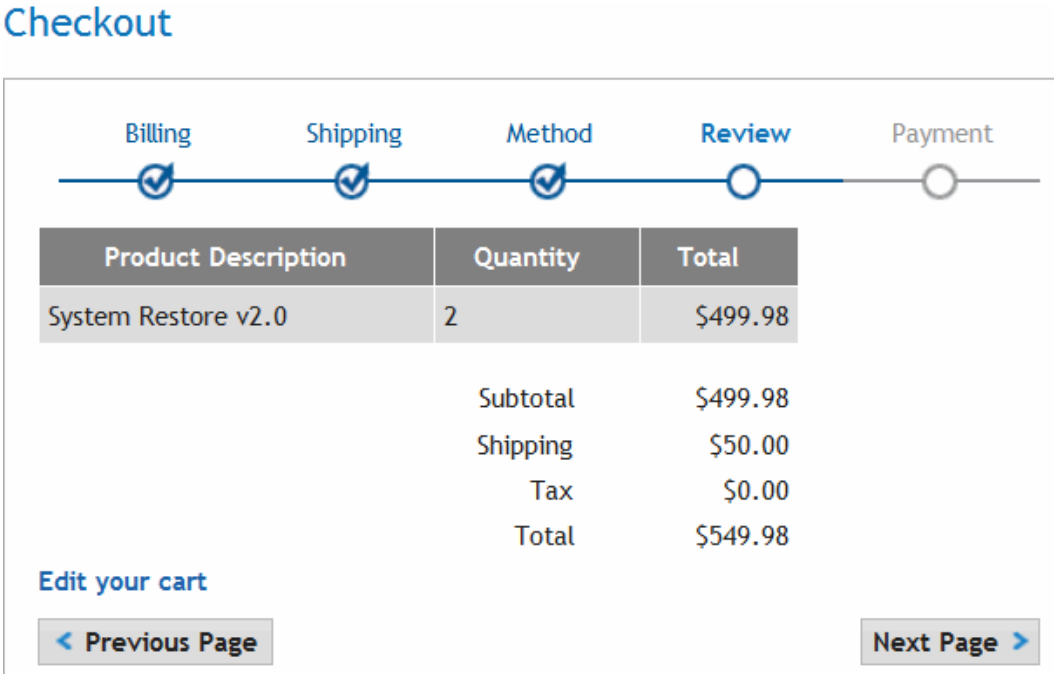
Select the type of shipping for products. Shipping methods that appear are defined in the **Workarea > Settings > Commerce > Shipping > Methods**. See also: Configuring Shipping Methods.



Review order

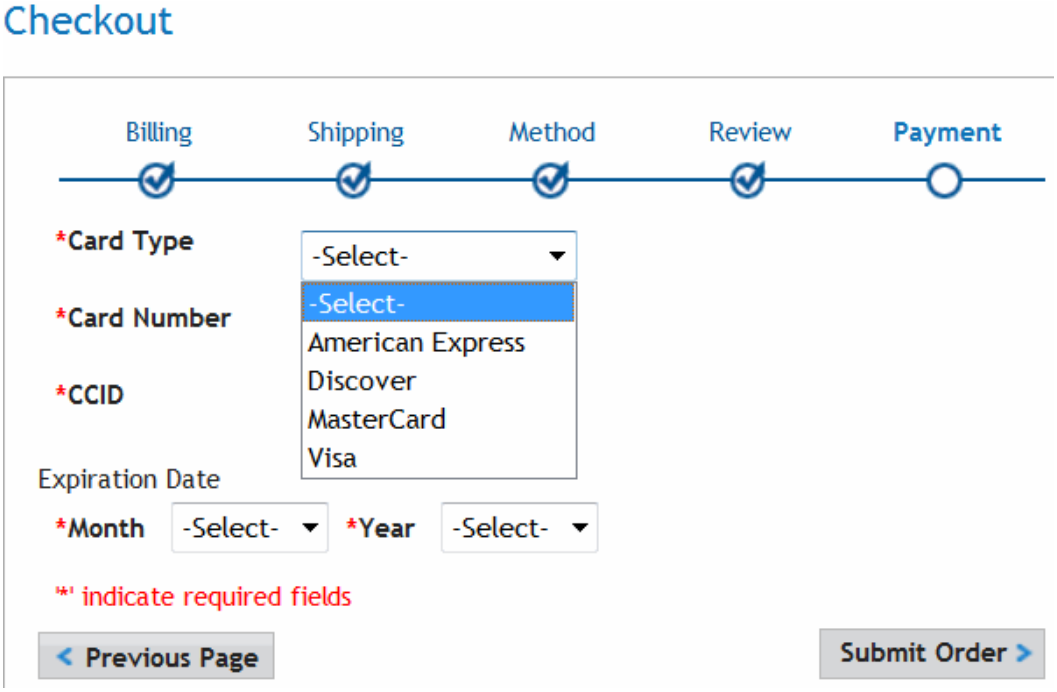
Displays information about products being purchased: their price, shipping charges, discounts, and taxes. At this point, if customers want to modify their cart, they click

Edit your cart. For example, a customer wants to apply a coupon to the cart. For this link to work properly, add the path to the template that contains the Cart server control to the Checkout server control's `TemplateCart` property.



Submit order

On the Submit Order page, a customer enters payment information. The customer can pay by check, credit card, or PayPal, depending on what has been set up in the eCommerce Payment Options screen. When a customer enters the information, the customer clicks **Submit**. At that point, the charge is submitted to the payment gateway, and the order is posted in Ektron. See also: Processing Orders.



Order complete

The Order Complete page displays a Thank You note, the order ID, and links to continue shopping or view an order's history.

The **Continue Shopping** link appears when you add the path of a template that allows a customer to find products to the `TemplateShopping` property. For example, you send them to a template containing a `ProductSearch` or `ProductList` server control.

The **Order History** link appears when you add the path of a template containing the OrderList server control to the `TemplateOrderHistory` property.

Adding a custom field to the checkout control

This example demonstrates how to add a custom field to the Checkout control's Billing Info entry screen.

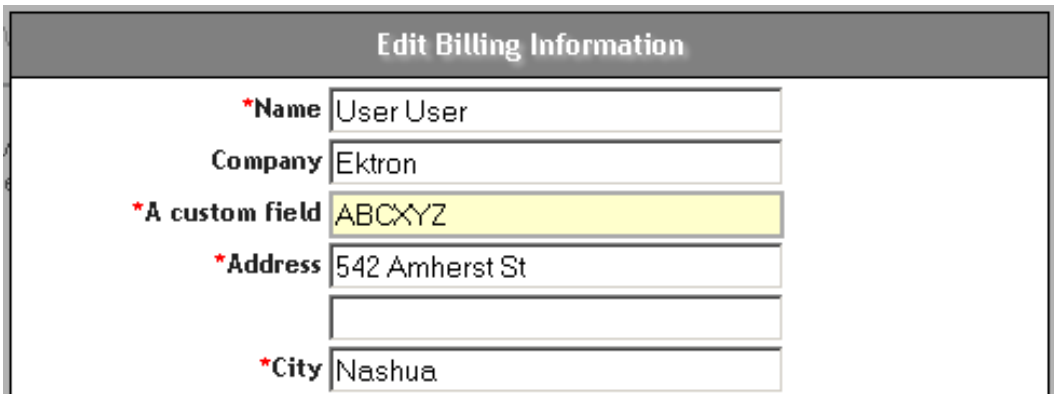
Step1: Add a field to the XSL

In the workarea/XSLT/Commerce/Checkout/ folder, look for the Checkout_ BillingInfo.xsl file. In the file, locate `EktronCheckout_BillingInfo_Company_`. Under the bulleted list item for company, add the following:

```
<li class="EktronCheckout_Required EktronCheckout_SerializableContainer">
      <label class="EktronCheckout_Row_LeftContents">
         <xsl:attribute name="for">
   EktronCheckout_BillingInfo_CustomField_<xsl:value-of select="$ControlId"/>
         </xsl:attribute>
        <span class="EktronCheckout_RequiredSymbol">*</span>
        A custom field
      </label>
      <input class="EktronCheckout_Company EktronCheckout_Row_RightContents">
         <xsl:attribute name="name">
   EktronCheckout_BillingInfo_CustomField_<xsl:value-of select="$ControlId"/>
         </xsl:attribute>
         <xsl:attribute name="id">
   EktronCheckout_BillingInfo_CustomField_<xsl:value-of select="$ControlId"/>
          </xsl:attribute>
         <xsl:attribute name="value">ABCXYZ</xsl:attribute>
      </input>
 </li>
```

Important notes:

- `EktronCheckout_Required`. Denotes that the field is required and will impose validation. `EktronCheckout_NotRequired` is also acceptable.
- `EktronCheckout_SerializableContainer`. Tells the checkout control to process and include the field in this <li> tag with the callback.
- `EktronCheckout_BillingInfo_CustomField_` . The name of the field, where `EktronCheckout_BillingInfo` is replaced with `param__` internally. The last portion, `CustomField`, is how the field will be referenced (see below).



Step2: Get the value of the field

In the codebehind of the page with the Checkout control, you need the following imports:

```
using Ektron.Cms.Controls;
using Ektron.Cms.Controls.ECommerce;
using Ektron.Cms.Controls.ECommerce.Checkout;
```

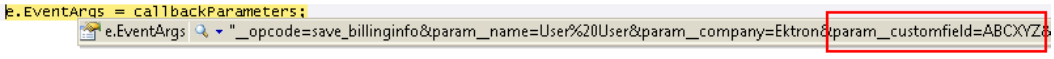
Next, place the following into the Page_Init method:

```
cmsCheckout.FilterAjaxCallback += this.HandleFilterAjaxCallback;
```

Then implement the HandleFilterAjaxCallback method:

```
protected void HandleFilterAjaxCallback(object sender,
FilterAjaxCallbackEventArgs e)
{
     string callbackParameters = e.EventArgs;
     System.Collections.Specialized.NameValueCollection data =
Ektron.Cms.Common.EkFunctions.ConvertToNameValueCollection(callbackParameters,
 true);
      string customFieldValue = data["param__customfield"];
}
```

Upon completing the Billing Info screen, the data is returned as part of the EventArgs, with the prefix `param__` (note the double underscore).



It can then be obtained directly by converting the string to a `NameValueCollection` (or other construct). Developers are free to call their own methods from here.

CurrencySelect

8.50 and higher

The CurrencySelect server control lets a customer select the monetary type he will use to make purchases. When a currency is selected, eCommerce server controls use it for that order. If the customer closes the browser, the currency needs to be selected again the next time the site is visited.

For a currency to appear in the CurrencySelect server control, enable it in the **Workarea > Settings > Commerce > Currencies** screen. See also: Configuring Currencies.

Inserting the CurrencySelect server control onto a page

PREREQUISITE

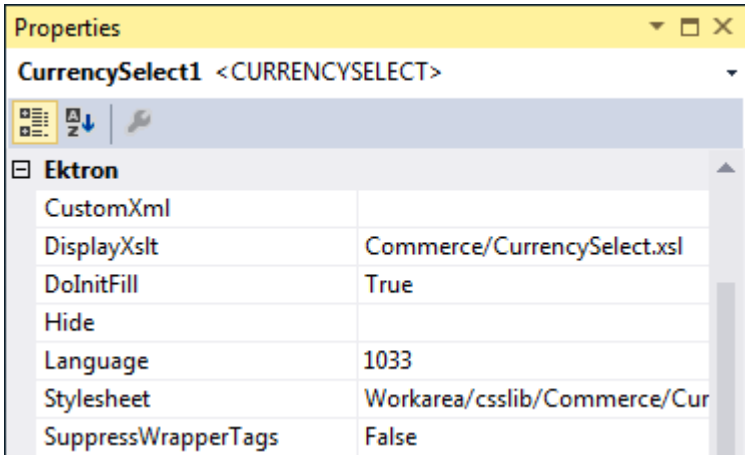
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **CurrencySelect** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:CurrencySelect ID="CurrencySelect1" runat="server" />
```

4. Click on `CurrencySelect` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



CurrencySelect properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CustomXml** (Code-behind Only) (String)

  Lets you inject custom XML into the generated XML before being processed by the XSLT. Enter a string of XML that you want to make available to the XSLT as follows:

  ```
  <root><customXml>custom-xml-inserted-here</customXml></root>
  ```

  See also: *Displaying custom XML in Ektron's server controls* on page 2162.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page. By default, the control uses *siteroot*\Workarea\Xslt\Commerce\CurrencySelect.xsl.

  WARNING! Files stored in the siteroot\Workarea folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the siteroot\workarea folder then edit it. If there is no default file, create the file outside the siteroot\workarea folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the siteroot\workarea folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

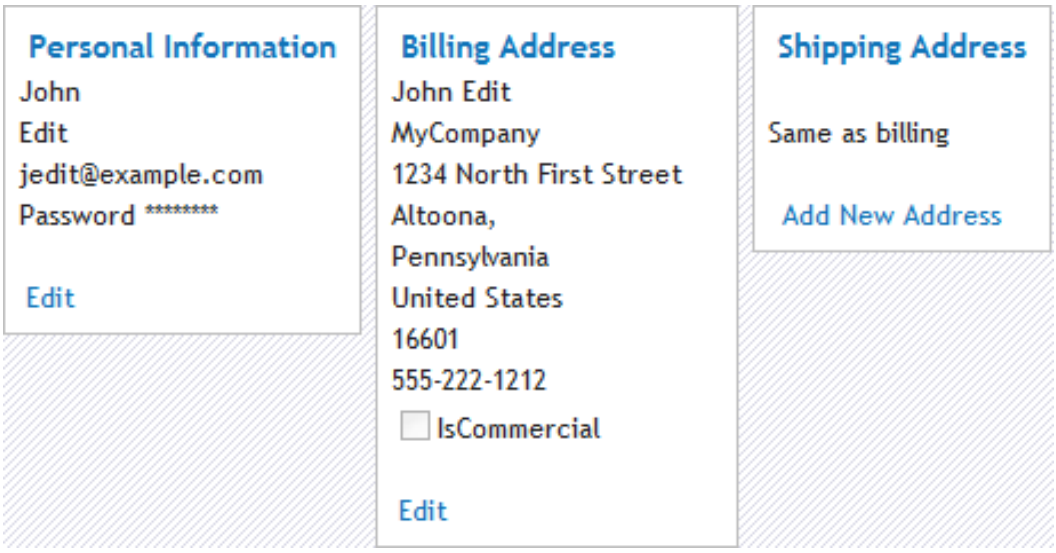
MyAccount

8.50 and higher

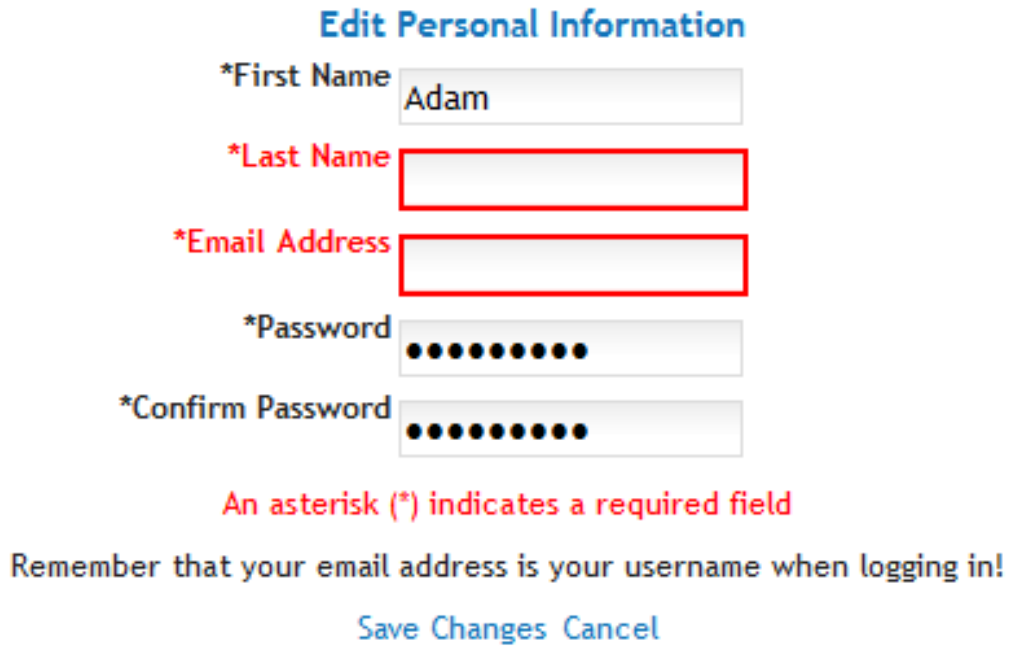
The MyAccount server control lets logged-in customers (site visitors) view and edit their Personal Information, Billing Address and Shipping Address. If the customer is not logged in and arrives at this control, one of the following happens:

- the visitor receives a message about not being logged in. This happens when the server control's `RedirectUrl` property is empty and the `web.config` file's `<add key="ek_RedirectToLoginURL" value="" />` value is blank.
- the visitor is redirected to a login page and once logged in, is sent back to the template containing the MyAccount server control. This happens when either the server control's `RedirectUrl` property contains the path of a template that contains a Login server control or when the `web.config` file's `<add key="ek_RedirectToLoginURL" value ="" />` `value` is set to the path of a template that contains a Login server control. If the control's `RedirectUrl` property and the `web.config` file contain different locations, the `RedirectUrl` property in the control overrides the `web.config` file.

Typically, the information displayed in this control is collected the first time a customer goes through the checkout process. A customer's information might appear in the Personal Information area before they have been through the checkout process if they are a current Ektron user or registered membership user.

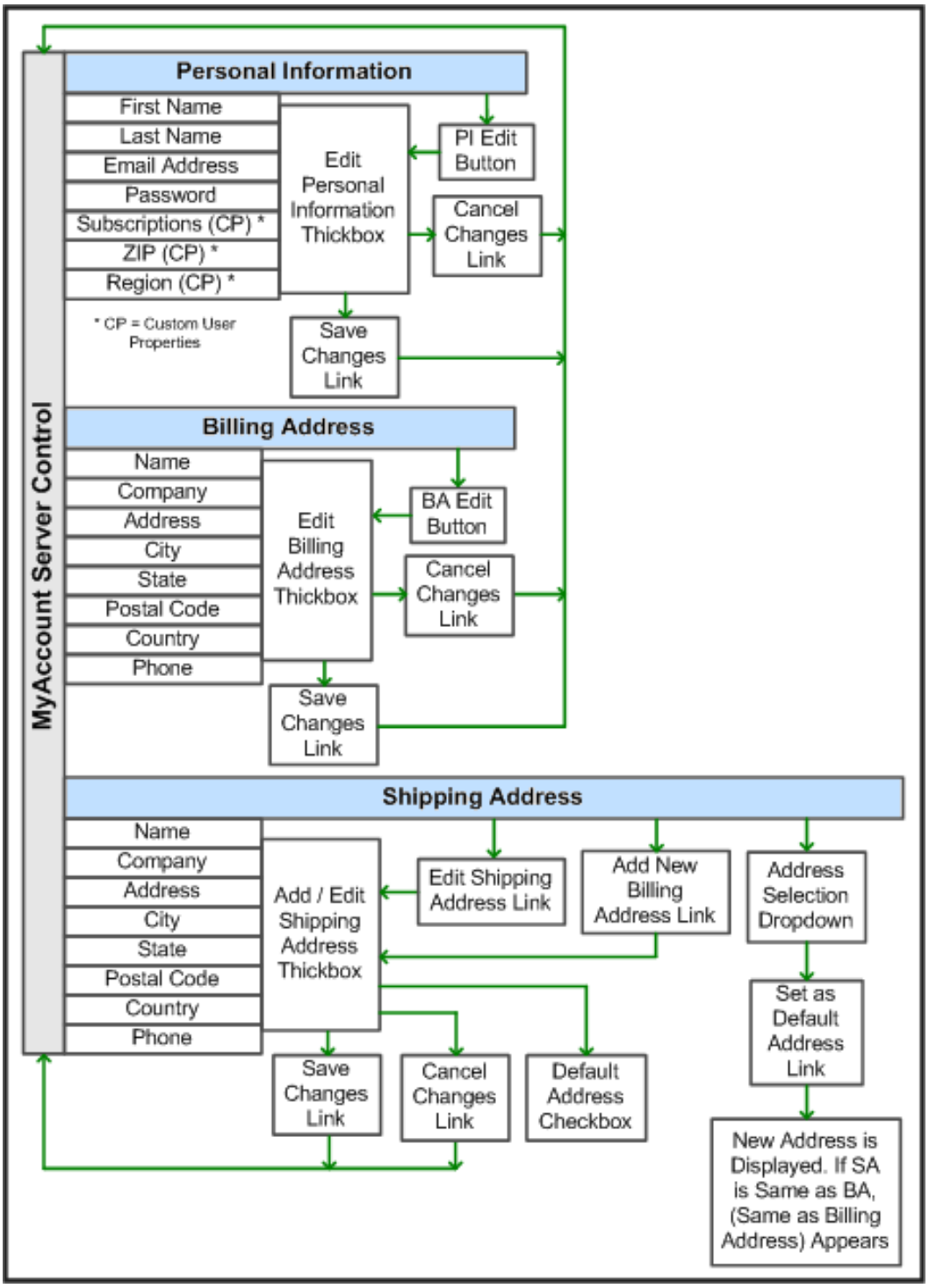


If a logged-in customer arrives at this server control and is missing required information, it must be entered before continuing. For example, a customer needs to provide a Last Name or E-mail Address.



As a developer, you need to create a link to this control in a site location that appears only after log in. For example, you might place the link on a menu or on a master page that appears after a customer logs in.

A customer typically arrives at this control by clicking a link placed on a main page, master page, or menu. The following figure shows the flow of the MyAccount server control.



Inserting the MyAccount server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

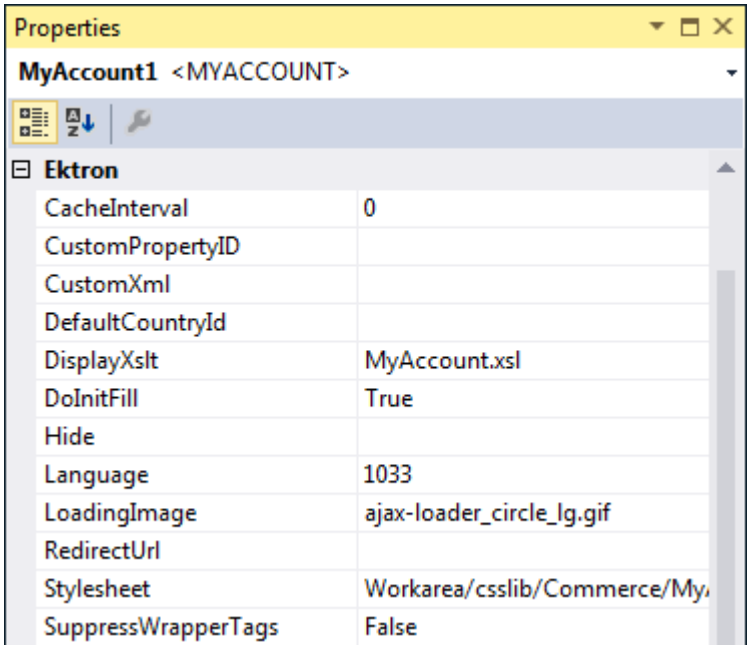
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.

3. Drag the **MyAccount** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:MyAccount ID="MyAccount1" runat="server" />
```

4. Click on `MyAccount` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



MyAccount properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **CustomerPropertyID** (String)

  An ID or comma separated list of IDs for a Custom User Property (or properties) to expose in the UI. These properties are used to define information about a user beyond the standard Ektron user properties; such as Username, First Name, Password and email Address.

  IDs for these properties can be found in the **Workarea > Settings > Configuration > Custom Properties** screen. See also: Custom User properties.

- **CustomXml** (Code-behind Only) (String)

  Lets you inject custom XML into the generated XML before being processed by the XSLT. Enter a string of XML that you want to make available to the XSLT as follows:

  ```
  <root><customXml>custom-xml-inserted-here</customXml></root>
  ```

  See also: *Displaying custom XML in Ektron's server controls* on page 2162.

- **DefaultCountryID** (Integer)

  The ID of the default country that appears in the Billing Address and Shipping Address. Set this ID to the country from which the majority of your customers will be making purchases. If the customer is from a different country, the customer can it change when editing the Billing or Shipping Address.

  NOTE: If a customer enters information in the Billing Address area, the Billing Country seeds the Shipping Country when adding a new Shipping Address.

  To find a country's numeric ID, sign into the Workarea. Then go to **Settings > Commerce > Configuration > Countries**. The Numeric ID is in the left column of that screen.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page. By default, the control uses `MyAccount.xsl`. This file is located in *`siteroot`*`\Workarea\Xslt\Commerce`.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LoadingImage** (String)

  The image to display while the control is fetching data. The default is *`siteroot`*`\Workarea\images\application\ajax-loader_circle_lg.gif`.

WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **RedirectUrl** (String)

  Enter the path of a template that contains the Login server control. When a customer arrives at this control and is not logged in, the control checks this property for a path to template containing a Login server control. If one exists, the customer is sent to that template. When the customer logs in, the template containing the MyAccount server control appears. If no path exists, the control checks the `web.config` file's <addkey="ek_RedirectToLoginURL"value=""/> value for a path.

- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

Editing personal information

The Personal Information area lets customers view and edit some information contained in their profile. By default, a customer can edit their First Name, Last Name, email Address, and Password. By adding a list of Custom User Property IDs to the `CustomPropertyID` property, you can let customers view and edit custom properties in their profile. See also: Custom User properties.

A customer can change the information in the Personal Information area by clicking **Edit** in the lower left corner of this area. The resulting page lets the person change any information in this area. When done, the customer clicks **Save Changes** to save the changes and return to the My Account page.

Editing a billing address

The Billing Address area lets customers view and edit their billing address information. A customer can edit their Name, Company, Address, City, State, Postal Code, Country, and Phone number. This information is used for billing purposes when the customer makes a purchase. Credit card companies typically want part or all of this information to match a customer's credit card account information when making a purchase.

To edit a customer's billing address, click **Edit** in the lower left corner of the Billing Address area.The resulting page let the person change any information in this area.

When done, a customer clicks **Save Changes** to save the changes and return to the My Account page.

Editing a shipping address

The Shipping Address area lets customers view, edit, delete or add a new address to their shipping address information. The fields in this area are the same as those in the Billing Address area with 2 exceptions: a **Same as Billing** check box, and a **Default Address** check box. The **Same as Billing Address** check box lets customers place a check mark in the box and use their billing address information as their shipping address. The **Default Address** check box lets customers specify the current shipping address as the default they want to use when making future purchases.

To edit a customer's shipping address, click **Edit** in the lower left corner of the Shipping Address area. The resulting page let the person change any information in this area. Note that a customer must uncheck **Same as Billing** to be able to edit the shipping address. Otherwise, it is locked to the billing address. When done, a customers clicks **Save Changes** to save the changes and return to the My Account page.

Adding a shipping address

To add a new shipping address, the customer clicks **Add New Address** located in the bottom center of the Shipping Address area.The resulting page lets the person enter new shipping address information. Note that the **Same as Billing** checkbox is not available when adding a new address. When done, a customer clicks **Save Changes** to save the changes and return to the My Account page.

Deleting a shipping address

To delete a shipping address, the customer clicks **Delete Address** at the bottom of the Shipping Address area. The visitor is asked to confirm. If the customer does and there are no other shipping addresses, the billing address is used as the shipping address. If the shipping address is the same as the billing address, you cannot delete it.

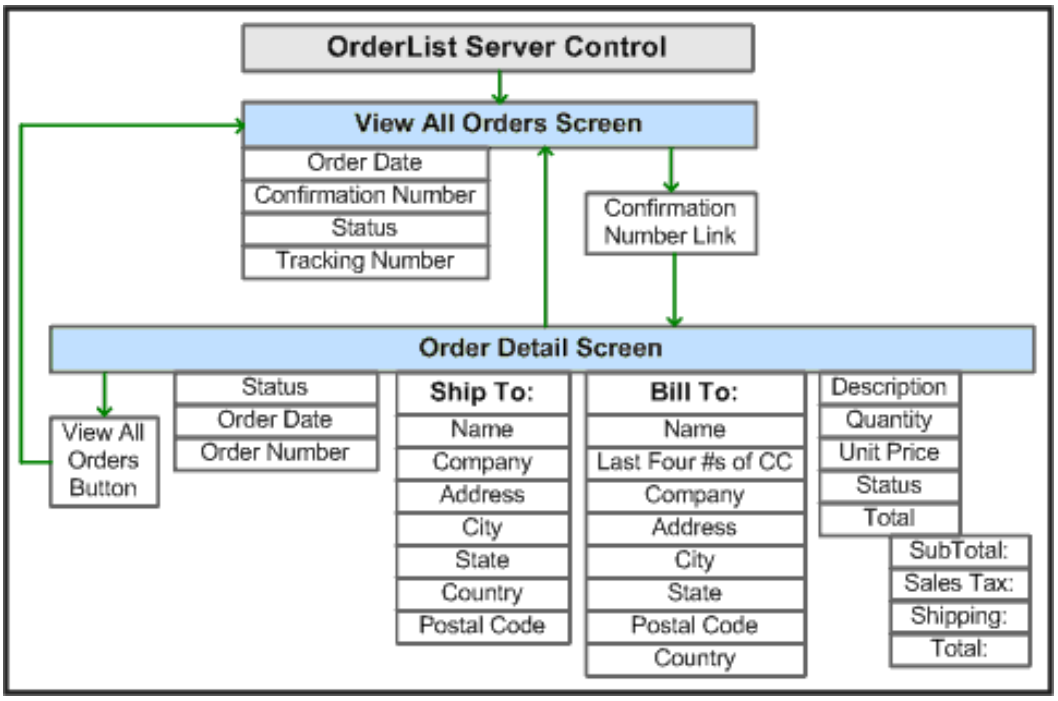
OrderList

8.50 and higher

The OrderList server control lets customers view a list of processed orders. Customers typically reach this server control by clicking a link on your site, such as the **View Order History** link.

Customers also can reach this control through a link at the end of the Checkout process. The link appears when the Checkout server control's `TemplateOrderHistory` property contains the path to the template containing the OrderList server control.

The following image depicts the flow of the OrderList server control.



Inserting the OrderList server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

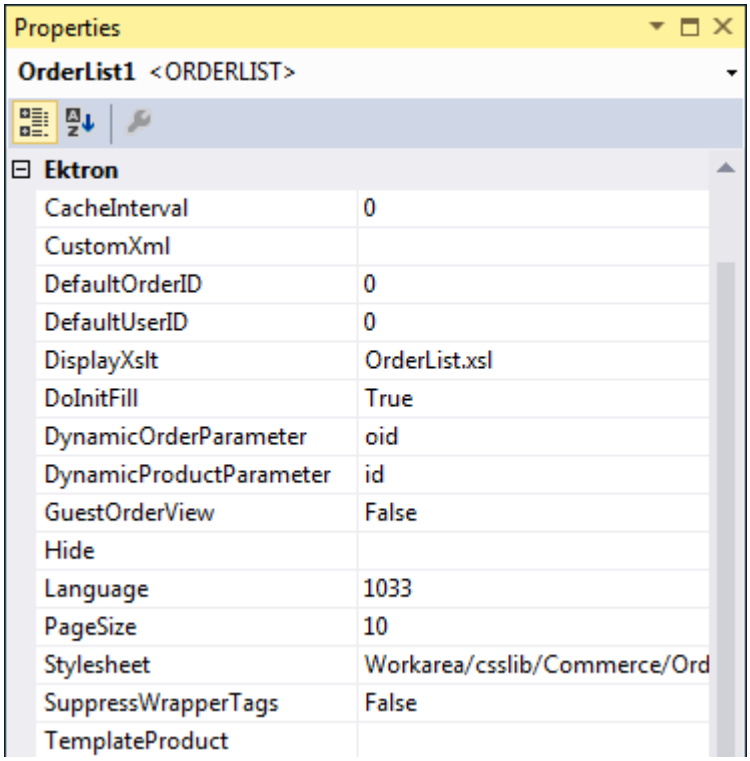
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **OrderList** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:OrderList ID="OrderList1" runat="server" />
   ```

4. Click on `OrderList` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



OrderList properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **CustomXml** (Code-behind Only) (String)
  Lets you inject custom XML into the generated XML before being processed by the XSLT. Enter a string of XML that you want to make available to the XSLT as follows:

  ```
  <root><customXml>custom-xml-inserted-here</customXml></root>
  ```

  See also: *Displaying custom XML in Ektron's server controls* on page 2162.
- **DefaultOrderId** (Long)
  The default ID of the list of orders to display.
- **DefaultUserId** (Long)
  The default ID of the user for which to display a list of orders.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page. By default, the control uses `OrderList.xsl`. This file is located in *`siteroot`*`\Workarea\Xslt\Commerce`.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicOrderParameter** (String)

  Sets the QueryString parameter name which is used to pass an order ID dynamically. By default, this parameter is OID.

- **DynamicProductParameter** (String)

  The QueryString parameter name which is used to pass the product ID to the product details page. For example, if your QueryString parameter for products is ID, enter that in this property. Then, when customers click a product in their order list, this parameter is passed with the product's ID to the product details page.

- **GuestOrderView** (Boolean)

  Set to **true** to allow guest account visitors to view their orders. If you do, a visitor can go to the View Orders screen, enter an order number and an email address, then view all submitted orders. See also: *Viewing a customer orders list* on the next page.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **PageSize** (Integer)

  The maximum number of orders to show per page.

- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder

outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TemplateProduct** (String)
  The URL path of the template that allows a customer to view the details of a product in an order. When a path is entered, the product's title becomes a link that lets customers navigate to the template containing details of the product; for example, the template containing a Product server control.

Viewing a customer orders list

When customers arrive at this server control, they see the View All Orders portion of the server control.

This part of the server control displays a list of submitted orders. Customers can view each order's date, confirmation number, and status. When the number of orders exceeds the number defined in the `MaxResults` property, the list is paged and a user can navigate among pages with the links provided.

A customer can click the confirmation number to view the Order Details screen.

Order Status

Status: OnHold
Order Date: Wednesday, October 01, 2008
Order Number: 3

View All Orders

Ship To:	Bill To:
Admin 542 Amherst Street (Route 101A) Nashua, New Hampshire United States 03063	Admin **********1111 542 Amherst Street (Route 101A) Nashua, New Hampshire United States 03063

Description	Qty	Unit Price	Status	Total
Atrium Lounge Chair	1	$300.00	OnHold	$270.00
			Subtotal:	**$270.00**
			Coupon Total:	($30.00)
			Tax:	$0.00
			Shipping:	$50.00
			Total:	**$320.00**

To navigate back to the list of all orders, customers click **View All Orders**.

Viewing order details

- **Status**. The status of the order.
- **Order Date**. The date the order was submitted. This is the date an order went through the checkout process.
- **Order Number**. The order's ID number.
- **Ship To:**. Information needed to ship the order to customer.
  - Person to whom the product is being shipped
  - Company Name (optional)
  - Street Address
  - City and State or Region
  - Country
  - Zip Code
  - Tracking Number. This field appears if a tracking number has been entered in the View Order screen in the Workarea.
- **Bill To:**. Displays information about who is being billed for the order.
  - Person paying for the order
  - Credit Card information. For security purposes, only the last 4 digits appear.
  - Company Name (optional)

  - Street Address
  - City and State or Region
  - Country
  - Zip Code
- **Description**. A list of products in the order. When the path to the product server control is entered in the `TemplateProduct` property, the product's title becomes a hyperlink.
- **Quantity**. The total number of each product.
- **Sale Price**. The per unit price of each product.
- **Total**. The total amount paid for each product ordered. This is the Unit Price multiplied by the Quantity.
- **Subtotal**. The sum of all the order's line products before additional charges or discounts, such as tax, shipping and coupons.
- **Coupon Total**. The total amount discounted, based on all applied coupons.
- **Tax**. The tax amount applied to this order. See also: Configuring eCommerce.
- **Shipping**. The amount charged to ship the order to the recipient. See also: Configuring eCommerce.
- **Total**. The sum of all charges, including all line items, tax, shipping and handling.

Product

8.50 and higher

The Product server control lets customers view the details of a product and add it to their cart. To display a product on a Web page, drag and drop a Product server control on a template and enter a product ID in the `DefaultProductID` property. You could also dynamically pass a product's ID to the control by setting `DynamicProductParameter` property to the `QueryString` parameter used for product IDs.

This control handles each class of product Ektron provides such as Products, Kits, Bundles or Subscriptions without having to make any adjustments to the control.

Customers typically reach this server control when they click a product from either the ProductSearch or ProductList server control. When customers clicks a product, title or image in either of these controls, they are forwarded to the Product server control.

In addition, customers can reach this control from the Cart server control. In that control, customers click a product's title and are taken to the Product server control.

When a customer has viewed the product and decided to purchase it, they click **Add to Cart** in the control.

Inserting the Product server control onto a page

PREREQUISITE

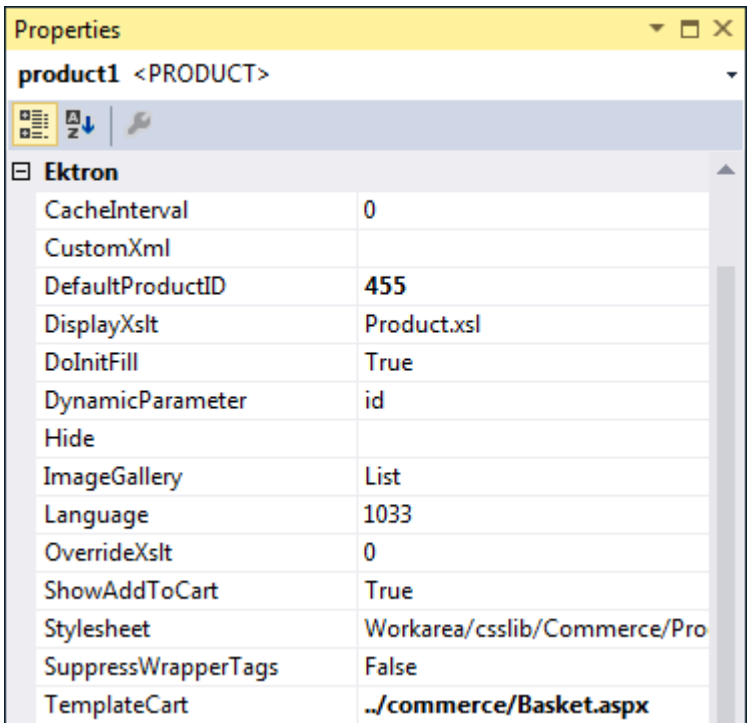
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Product** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Product ID="Product1" runat="server" />
```

4. Click on `Product` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



Product properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **CustomXml** (Code-behind Only) (String)

  Lets you inject custom XML into the generated XML before being processed by the XSLT. Enter a string of XML that you want to make available to the XSLT as follows:

  ```
  <root><customXml>custom-xml-inserted-here</customXml></root>
  ```

  See also: *Displaying custom XML in Ektron's server controls* on page 2162.

- **DefaultProductID** (Integer)

  Enter a default product's ID. This is the default product that's displayed when the template containing this control is viewed by a customer and a product ID is not dynamically passed. To make this server control dynamic, enter zero (0) for this property and set the `DynamicProductParameter` to the QueryString parameter used to pass the product ID.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page. By default, the control uses `product.xsl`. This file is located in *`siteroot`*`\Workarea\Xslt\Commerce`. See also: *The OverrideXslt and DisplayXslt properties* on the facing page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  The QueryString parameter name, which is used to read the product ID. For example, if your QueryString parameter for products is ID, enter that. Then, when customers view a product's details, the product's ID is passed to this control.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **ImageGallery** (ImageGalleryType)

  Determine if the image gallery appears along with a product's information. Images that appear in the gallery are set in the product's Media tab when creating or editing a product. The Gallery Display property for each image on the Media tab must be set to Yes. Choices:

- **List**. Display gallery images
- **None. H**ide gallery images

In the Image Gallery, images are displayed at their smallest size. If a customer clicks an image, a full size version appears.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **OverrideXslt** (Integer)

  Specifies an XSLT identified in the Edit Product Type Configuration screen.

- **ShowAddToCart** (Boolean)

  Set to true if you want to the **Add to Cart** button to appear. The default is True. Setting this property to false lets you show details of a product, but not offer it for sale. For example, a product is no longer for sale, but you want to allow people who purchased it to view details. Also, by using code-behind to dynamically set the property, you could create code that checks your inventory system and hides the button if a product is out of stock. You could also accomplish this for a product by removing the check from the **Buyable** property when creating or editing a product in the Workarea.

- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

- **TemplateCart** (String)

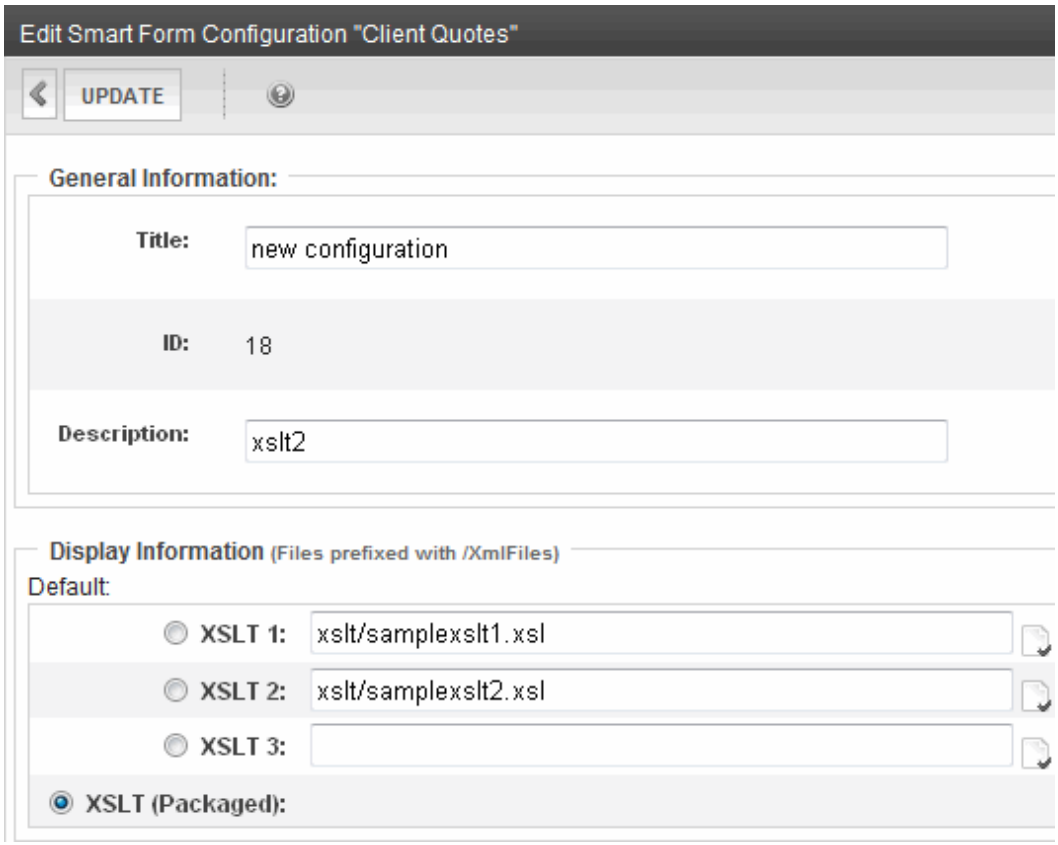
  Enter the URL path of the template that contains the *Cart* on page 2359 server control. The path can be relative or absolute.

The OverrideXslt and DisplayXslt properties

The `DisplayXslt` property is optional. If used, it specifies an external XSLT file. If the `DisplayXslt` property is not defined, the `OverrideXslt` property specifies an XSLT identified in the Edit Product Type screen. The following list provides more information about these properties.

- The default XSLT is specified in the Edit Smart Form Configuration screen. To use this display XSLT, enter 0 for OverrideXslt.

  NOTE: In this example, the XSLT Packaged option is the default XSLT, since it is selected. XSLT Packaged is the XSLT from the Edit Smart Form Configuration screen (that is, the XSLT created in the Data Designer).



- **XSLT 1** from the Edit Smart Form Configuration screen; to use this display XSLT, enter 1 for OverrideXslt.
- **XSLT 2** from the Edit Smart Form Configuration screen; to use this display XSLT, enter 2 for OverrideXslt.
- **XSLT 3** from the Edit Smart Form Configuration screen; to use this display XSLT, enter 3 for OverrideXslt.
- An absolute or relative path to an XSLT file. For DisplayXslt, enter an external XSLT file not specified in the Edit Product Type Configuration screen. For example: `sample.xslt`. No value is required for OverrideXslt. If an XSLT value exists, **OverrideXSLT** is ignored.

IMPORTANT: Files stored in the *siteroot*`\Workarea` folder are overwritten (or deleted) when you upgrade. To avoid problems, back up the files to a folder outside the `siteroot\Workarea` folder.

Displaying a product

A *product* is an item that has no kit, bundle or subscription information associated with it.

When displaying a simple product, the Product server control displays the following information:

- **Title** (optional). The **Title** field in the product's Smart Form. A content editor enters this information when creating a product in the Workarea. This area does not use the title of the product in Ektron's Workarea.
- **Description** (optional). The Description field in the product's Smart form. A content editor enters this information when creating a product in the Workarea.
- **Image Gallery**. The Media tab when creating or editing a product. The Gallery Display property for each image on the Media tab must be set to Yes. In the Image Gallery, images are displayed at their smallest size. When a customer clicks an image, the full size version appears. See also: *ActiveTopics* on page 2165.
- **Price**. The Pricing tab when creating or editing a product. This is the price defined in the **Our Sales Price** area. This is not the list price. See also: *ActiveTopics* on page 2165.
- **Add to Cart**. This button appears in the server control when:
  - a path to the Cart server control is defined in the `Templatecart` property
  - the product is buyable. When a product is not buyable, information about the product appears, but customers cannot add it to their cart. This property is set in the Workarea when creating or editing a product.
  - the `ShowAddToCart` property is set to True

  When customers click this button, the product is added to their cart and they are sent to a template containing the Cart server control.

You can hide this button by setting the `ShowAddToCart` property to false. This lets you show details of a product, but not offer it for sale. For example, you have a product that is no longer for sale, but you want to allow people who purchased the product to view its details.

Also, by using code-behind to dynamically set the property, you could create code that looks at your inventory system and hides the button depending on whether a product is in stock.

Using the Add to Cart button with aliasing

When a product has an alias path associated with it:

- Make sure the `TemplateCart` property's path is relative to the site root. For example:

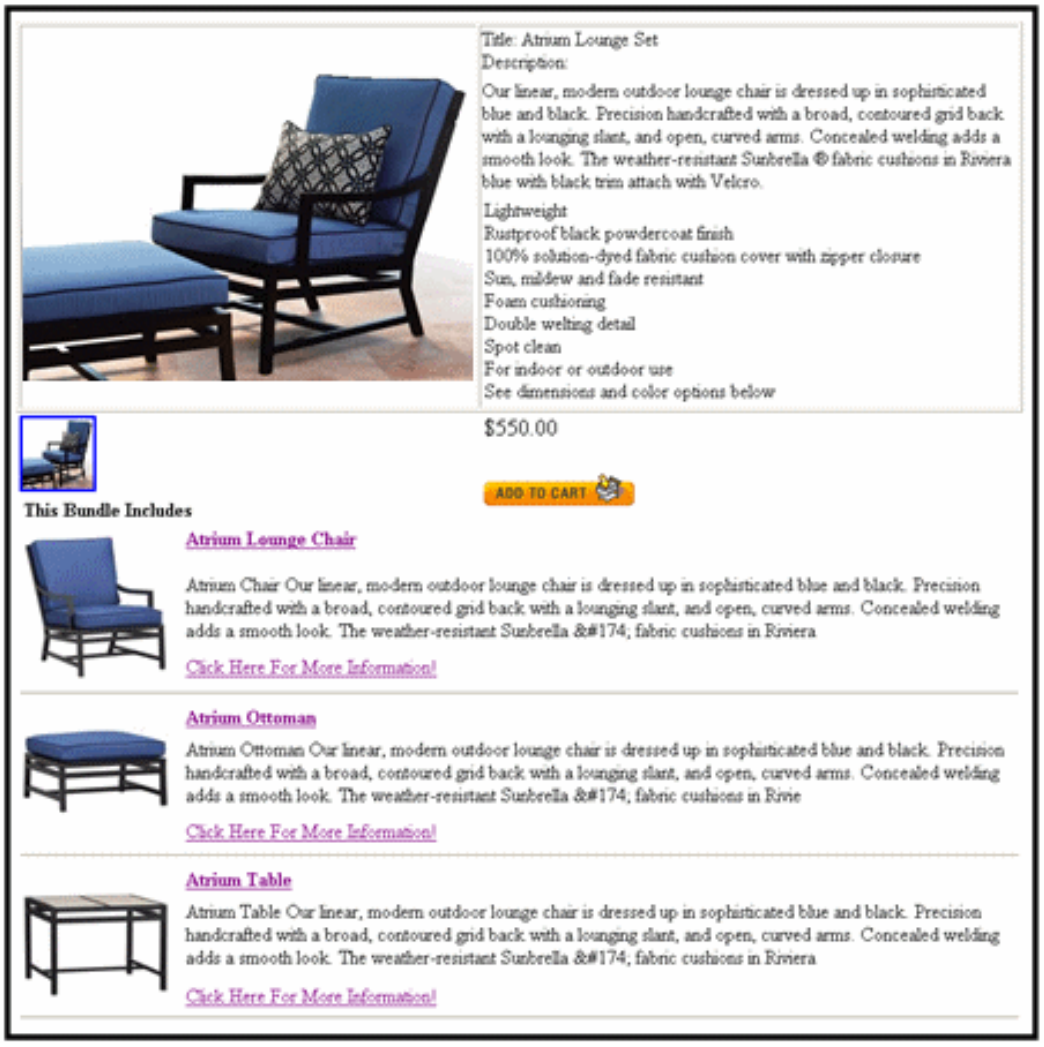
```
TemplateCart="Developer/Commerce/CartDemo.aspx"
```

- Add the following to the code-behind page. This example is in C#:

```
protected void Page_Init(object sender, EventArgs e)
{ Utilities.RegisterBaseUrl(this.Page); }
```

Displaying a bundled product

A bundled product is made up of multiple products that have been grouped together for sale as one product.



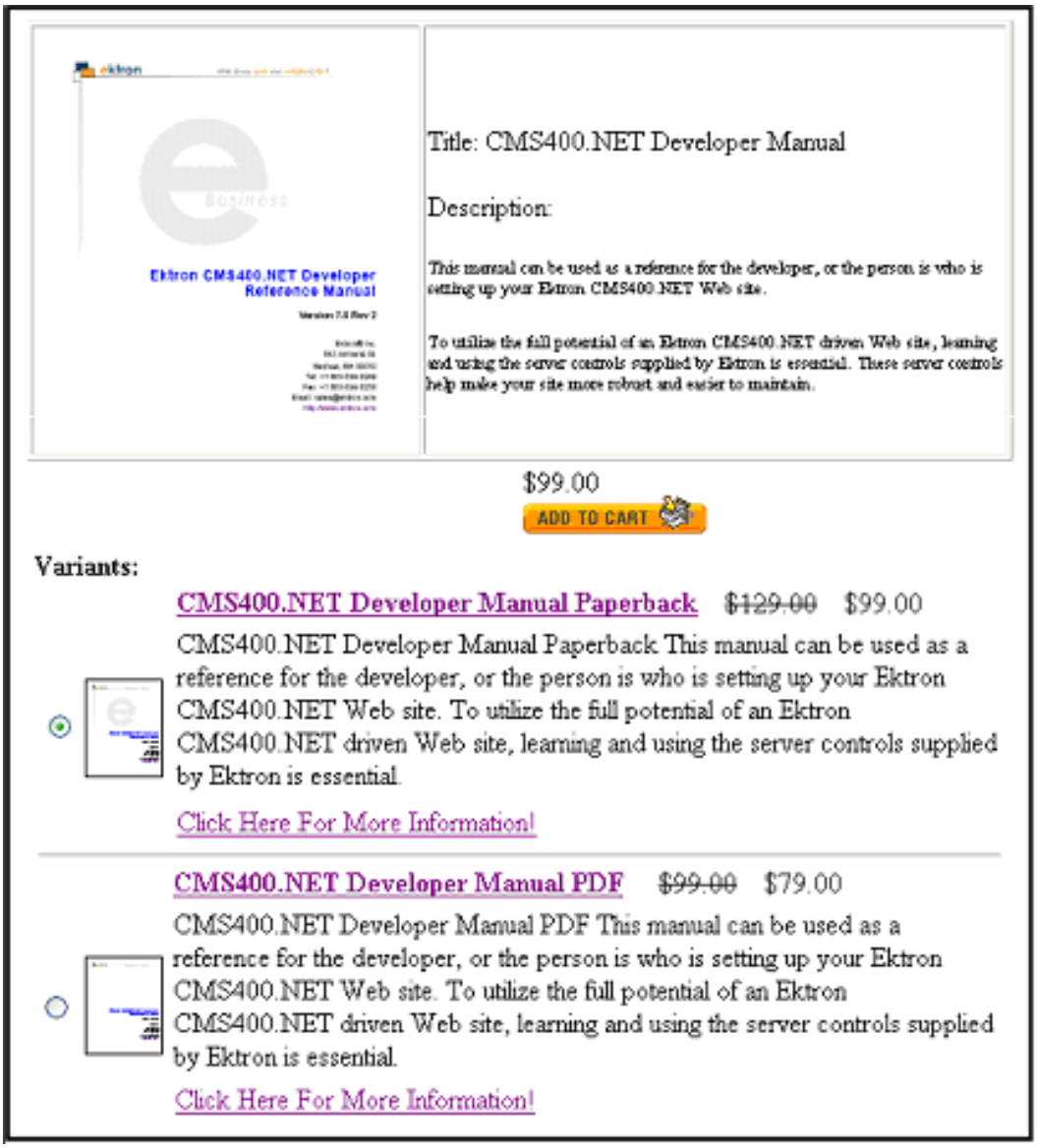
When displaying a bundled product, the Product server control displays all information displayed in a Product as well as information about the individual components of the bundle.

The **This Bundle Includes** area includes products listed on the **Items** tab for a Product Bundle in the Workarea. A content editor adds existing products to this tab when creating the bundle.

Any products on the tab are displayed with the image, title and description for each product. A link to additional information about each product is also displayed.

Displaying a complex product

A Complex Product lets the customer choose between variations of a product. For example, if your site sells books, variant selections might be Paperback or Electronic.

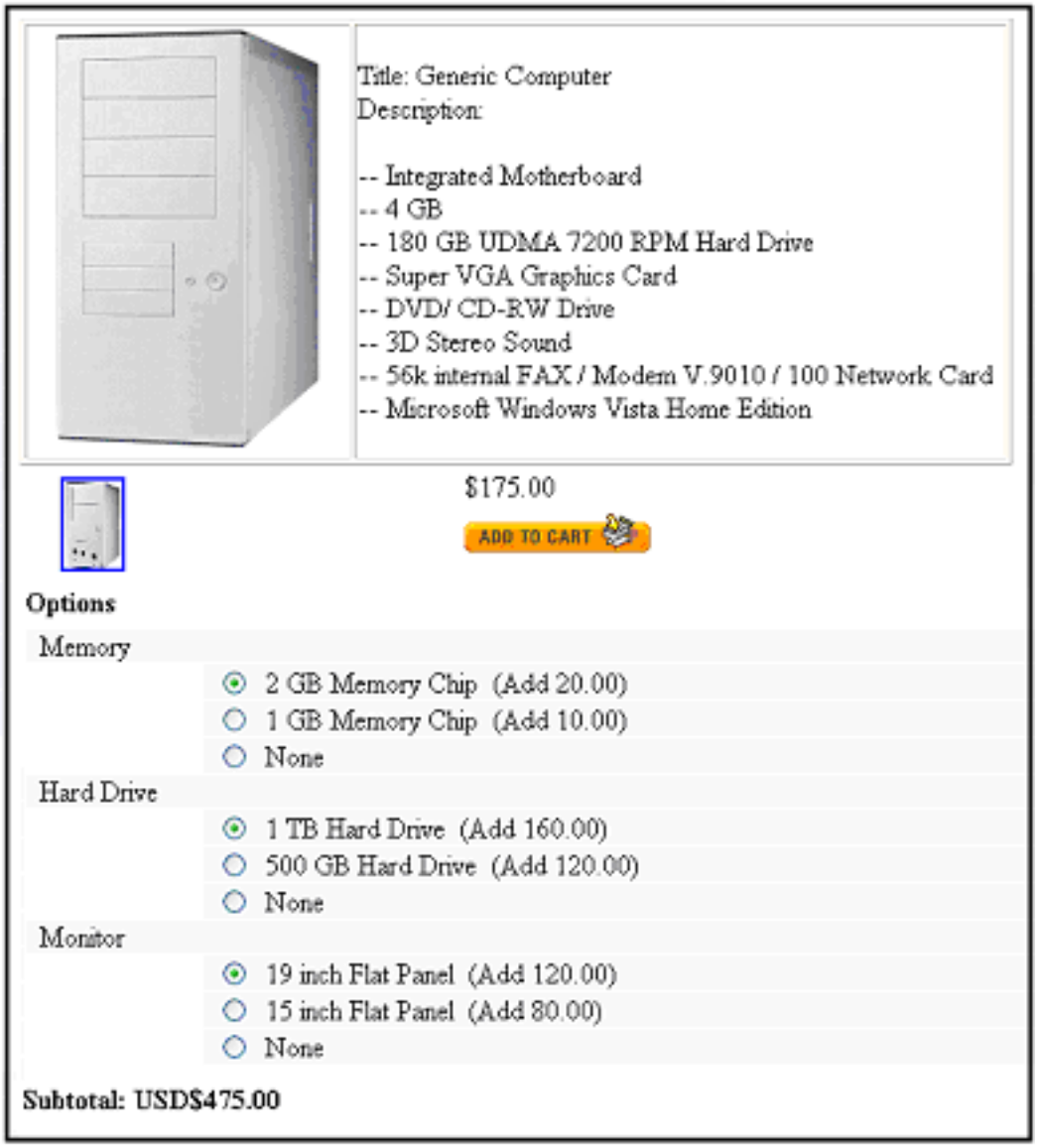


When displaying a Complex Product, the Product server control displays all of the information displayed in a Product in addition to information on product variants. The **Variants** area includes products listed on the **Items** tab. A content editor adds products to this tab when creating content.

Products on the **Items** tab are displayed with a radio button, image, title and description. A link to additional information about each product is also displayed. The radio buttons are used to select which product will be added to the cart.

Displaying a kit

A kit lets the customer select product options, which can affect the product's price. There is no limit to the types of options you can add, nor to the number of items in an option. For example, a customer purchasing a computer can add RAM, a hard drive, and a larger monitor.



When displaying a kit, the Product server control displays all information displayed in a product in addition to options and subtotal information.

The **Options** area displays product options based on the Item tab for a kit in the Workarea. Options are divided into groups. A radio button, name and price appears for each item. The radio button lets you select one item from each group.

The **Subtotal** area shows the updated cost of the product with all options.

ProductList

8.50 and higher

The ProductList server control displays a list of products on a Web page.

NOTE: Private catalog entries appear in display of the Product List server control only if the user is logged in and has at least Read-Only permissions for its catalog folder. Making Content Private

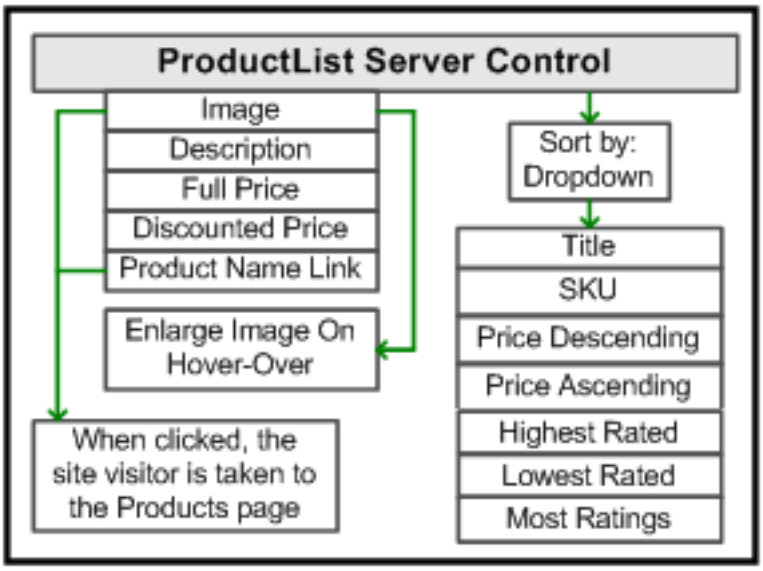
You decide which products appear by selecting a `SourceType` and populating either the `SourceId` or the `IdList` property, depending on the source type. You can choose from these source types.

If you want to display	Set the SourceType property to	In the `SourceId` property, enter	In the `IdList` property, enter
All products for a selected catalog	Catalog	The ID of the catalog	
All products across multiple catalogs	CatalogList		A comma separated list of catalog IDs
All products for a selected taxonomy	Taxonomy	The ID of the Taxonomy	
All products across multiple taxonomies	TaxonomyList		A comma separated list of taxonomy IDs
All products for a selected collection	Collection	The ID of the Collection	

If you want to display	Set the SourceType property to	In the `SourceId` property, enter	In the `IdList` property, enter
Display a list of products	IdList		A comma separated list of product IDs

There are several ways customers might arrive at the ProductList server control, such as

- when a list of products appears on the side of a page
- through a link in a master page
- when they click **Continue Shopping** on the Cart or Checkout server control.



Inserting the ProductList server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

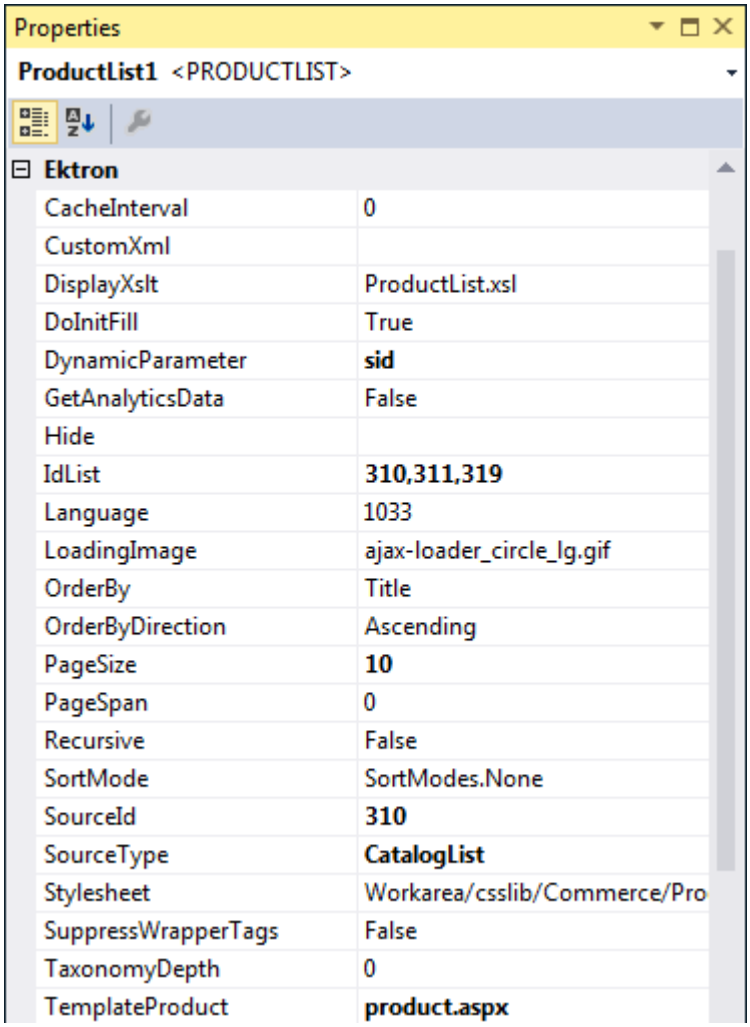
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **ProductList** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:ProductList ID="ProductList1" runat="server" />
   ```

4. Click on `ProductList` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



ProductList properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **CustomXml** (Code-behind Only) (String)

  Lets you inject custom XML into the generated XML before being processed by the XSLT. Enter a string of XML that you want to make available to the XSLT as follows:

  ```
  <root><customXml>custom-xml-inserted-here</customXml></root>
  ```

  See also: *Displaying custom XML in Ektron's server controls* on page 2162.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page. By default, the control uses `ProductList.xsl`. This file is located in *siteroot*`\Workarea\Xslt\Commerce`.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

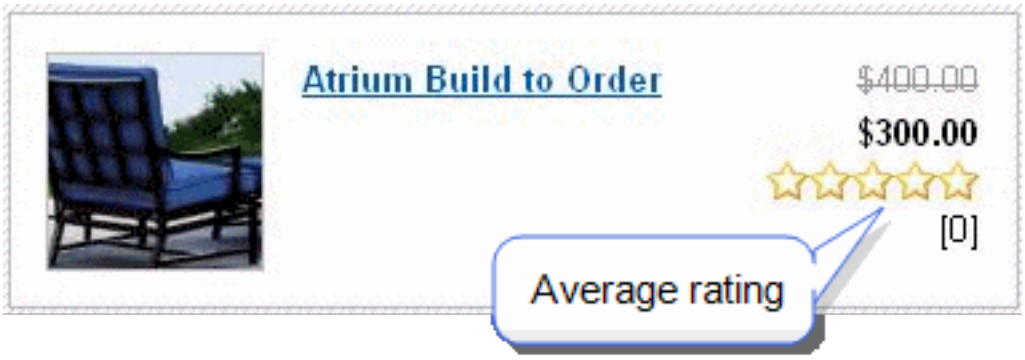
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Sets the QueryString parameter to read a source's ID dynamically. For example, if your QueryString parameter for a source ID is SID, enter that.

- **GetAnalyticsData** (Boolean)

  Set to `true` if you want the average rating for this product to appear with the product display, as shown below.

  Below the rating, a number indicates how many customers have rated the product. For information on customer product ratings, see *ContentReview* on page 2342.

  IMPORTANT: This property provides reliable data only when the Business Analytics feature is on.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **IdList** (String)

  Enter a comma-separated list of IDs if the `SourceType` property is set to IdList, TaxonomyList or CatalogList.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LoadingImage** (String)

  The image to display while the Product List is loading. The default is *siteroot*`\Workarea\images\application\ajaxloader_circle_lg.gif`.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **OrderBy**

  (*SortMode* property must be set to **None**)

  For further definition of the values shown here, see the **entryPropertyenumeration** in the **API documentation**.

  - **AverageRating**
  - **CatalogId**
  - **CollItemsDisplayOrder** (only available if `SourceType` property = Collection)
  - **ContentStatus**
  - **CurrencyId**
  - **EndDate**
  - **EntryType**
  - **GoLIve**
  - **Html**
  - **Id**
  - **IsArchived**
  - **IsBuyable**
  - **IsPublished**
  - **LanguageId**
  - **LastEditDate**
  - **LastEditorFirstName**
  - **LastEditorLastName**
  - **ListPrice**
  - **Media**
  - **NumberRated**
  - **ProductTypeId**

- **SalesPrice**
- **Sku**
- **Status**
- **Summary**
- **TaxClassId**
- **TaxItemsDisplayOrder** (only available if `SourceType` property = Taxonomy)
- **Title**
- **ViewCount**

- **OrderByDirection** (String)
  Select the direction of the `OrderBy` property. Choose Ascending or Descending.
- **PageSize** (Integer)
  Specify the number of items to show per page when a customer views the product list. If the number of items exceeds the quantity defined in this property, the list is paged and the customer can use the paging navigation system to move through the list.
- **PageSpan** (Integer)
  The number of pages to show before and after the current page. Enter zero to show all pages. For example, if you set this property to 2 and you are on page four of the product list, you see:

- **SortMode** (SortModes)
  Select the default way the product list is sorted when a customer first views the page. When the page is loaded, a customer can change the sort via a drop-down list. Choices are:
  - **Title**. Sorts in alphabetical order.

  - **SKU**. Sorts by the product number. This number is typically a unique number supplied by the producer of the product.
  - **Price Descending**. Sorts by price from highest to lowest.

- **Price Ascending**. Sorts by price from lowest to highest.
- **Highest Rated**. Sorts by rating from highest to lowest.
- **Lowest Rated**. Sorts by rating from lowest to highest.
- **Most Rated**. Sorts by products that have most ratings from highest to lowest.
- **None**. Use this setting to use *OrderBy* property.

See also: *Sorting the product List* on the facing page.

- **SourceId** (Integer)

  The ID of the catalog, collection or taxonomy that is being used as a product list. This property is used when the `SourceType` property is set to **Catalog**, **Collection** or **Taxonomy**.

- **SourceType** (SourceObjectType)

  The type of source being used to create the list. Choices are:

  - **Catalog**. Use the `SourceId` property to specify the ID of a single catalog when this source type is selected.
  - **CatalogList**. Use the `IdList` property to specify a list of catalog IDs when this source type is selected.
  - **Taxonomy**. Use the `SourceId` property to specify the ID of the single taxonomy when this source type is selected.
  - **TaxonomyList**. Use `IdList` property to specify a list of taxonomy IDs when this type is selected.
  - **IdList**. Use the `IdList` property to specify a list of product IDs when this source type is selected.
  - **Collection**. Use the `SourceId` property to specify the ID of a collection when this source type is selected.

- **Stylesheet** (String)

  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

- **TaxonomyDepth** (Integer)

  Enter the number of taxonomy levels to retrieve below each taxonomy/category. Only the top level appears in the control. The rest of the levels are available through XML in Code-behind. For example, if the taxonomy is **DVDs > Movies > Comedies**, and you set **Taxonomy Depth** to **2**, only DVDs and Movies are available in Code-behind.

To retrieve all categories for a taxonomy recursively, enter -1. A depth greater than 1 or using -1 is only useful if you create a custom output using the Taxonomy's XML. The default value is 1.

For a live site, Ektron strongly recommends leaving this value at **1**. Increasing this value can slow down your live Web server. However, for testing on a staging server, you can increase the depth.

- **TemplateProduct** (String)
  Specify the URL that contains the Product server control. This allows a customer to see the details of the product when the product link is clicked.

Sorting the product List

The ProductList server control lets a customer sort by:

- Title
- SKU
- Price High to Low
- Price Low to High
- Highest Rated
- Lowest Rated and Most Ratings

You can set the default sort order by setting the `SortMode` property.

For the Highest Rated, Lowest Rated and Most Ratings sorting options to work as intended, a ContentReview server control should be associated with each product. This lets customers rate your products.

For example, place a ContentReview control on the Master page of the template that display products, and set its `DynamicParameter` property to `ID`. Then, when customers view the product, they can rate and comment on it.

Recommendation

8.50 and higher

PREREQUISITE

Cross sell or upsell products have been assigned to a catalog entry.

The Recommendation server control lets you cross-sell or up-sell products to a customer based on another product's ID. After you create a product in the Workarea, you can define a list of other products that you want to associate with the new product. While viewing a product in the Workarea, choose **View > Cross Sell** or **Up sell**. Then, by adding the Recommendation server control to a Web form that contains a Product server control, the customer sees the recommendations when viewing the original product's details.

Typically, this control appears on a page along with a Product server control. By using this control in conjunction with the Product control, a customer can view the

details of a product and also receive suggestions on additional purchases. A customer can click the title to view the suggested product to view its details.

For example, if your site is selling a hat, mitten and scarf set, you might use this server control to cross-sell winter jackets. You could also use the control to up-sell a more expensive hat, mitten and scarf set, or a set that includes additional items.

The `RecommendationType` property determines whether the Recommendation server control is used for up-selling or cross-selling.

Inserting the Recommendation server control onto a page

PREREQUISITE

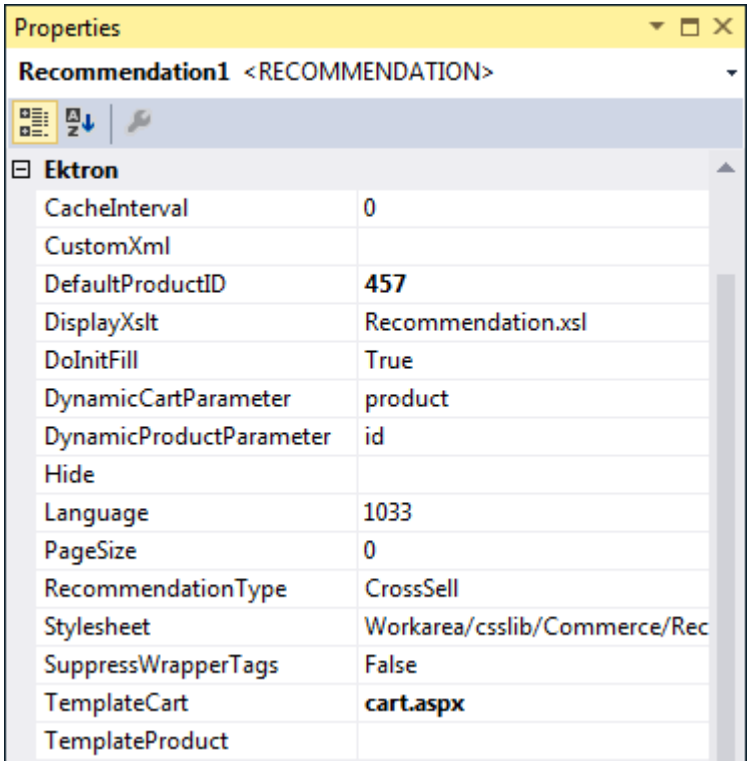
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Recommendation** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Recommendation ID="Recommendation1" runat="server" />
```

4. Click on `Recommendation` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



Recommendation properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **CustomXml** (Code-behind Only) (String)

  Lets you inject custom XML into the generated XML before being processed by the XSLT. Enter a string of XML that you want to make available to the XSLT as follows:

  ```
  <root><customXml>custom-xml-inserted-here</customXml></root>
  ```

  See also: *Displaying custom XML in Ektron's server controls* on page 2162.

- **DefaultProductID** (Integer)

  Enter a default product ID that contains either cross sell or up sell products. To make the server control dynamic, enter zero (0) in this property and set the `DynamicProductParameter` to QueryString parameter used to pass the product's ID.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page. By default, the control uses `Recommendation.xsl`. This file is located in *`siteroot`*`\Workarea\Xslt\Commerce`.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicCartParameter** (String)

  The QueryString parameter name which is used to pass the product ID to the Cart server control. For example, if your QueryString parameter for products is Product, enter that in this property. Then, when customers click **Add to Cart**, this parameter is passed with the product's ID Web form containing the *Cart* on page 2359 server control.

- **DynamicProductParameter** (String)

  The QueryString parameter name which is used to pass the product ID to the product details page. For example, if your QueryString parameter for products is ID, enter that in this property. Then, when customers click a product's title, this parameter is passed with the product's ID to the product details page.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

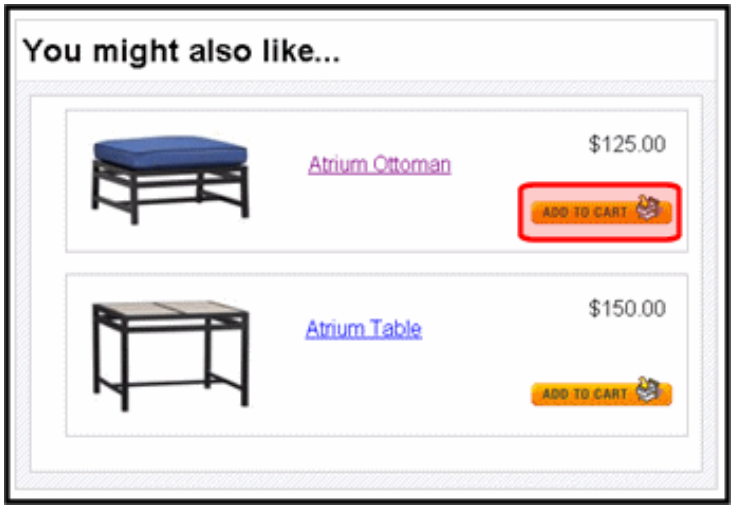
  If the template on which this server control resides includes a language selection control, and you want to let the customer select the language, enter zero (0).

  Otherwise, click the field, then the **ellipsis** button () and a popup box appears. Select a language from the list. This property shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **PageSize** (Integer)
  Specify the number of items to show per page when a customer views recommendations. If the number of items exceeds the quantity defined in this property, the list is paged and the customer can use the paging navigation system to move through the list.
- **RecommendationType** (`EkEnumeration.RecommendationType`)
  The type of recommendation to show. Choices are:
  - **CrossSell**. Related products that a customer might be interested in. For example, if you are selling Denim Jackets, you might want to cross sell them pants or shirts.
  - **UpSell**. Products that improve on the product being purchased. For example, if you are selling 7 megapixel digital cameras, you show a 10 megapixel digital camera.
- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.
- **SuppressWrapperTags** (Boolean)
- Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TemplateCart** (String)
  The URL to navigate to when a customer clicks **Add to Cart**. The path can be relative or absolute. When a path is entered, the **Add to Cart** button appears next to the product and allows a customer to add the product directly to their cart.
- **TemplateProduct** (String)
  Specify the URL that contains the Product server control. This allows a customer to see the details of the product when the product link is clicked. The path can be relative or absolute.

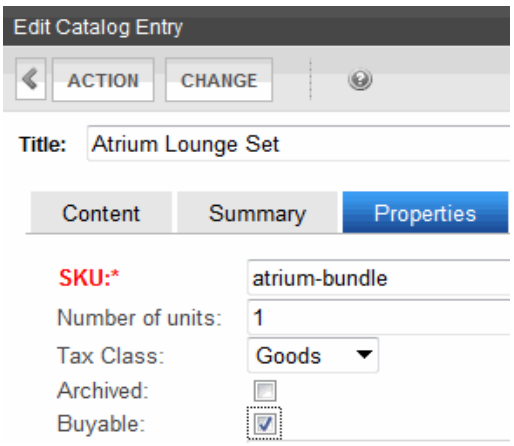
Enabling the Add to Cart button

Customers can add a product to their cart directly from the Recommendation server control by clicking **Add to cart** next to a product. This link appears below the price and allows them to skip the product's information page and add the product directly to their cart.

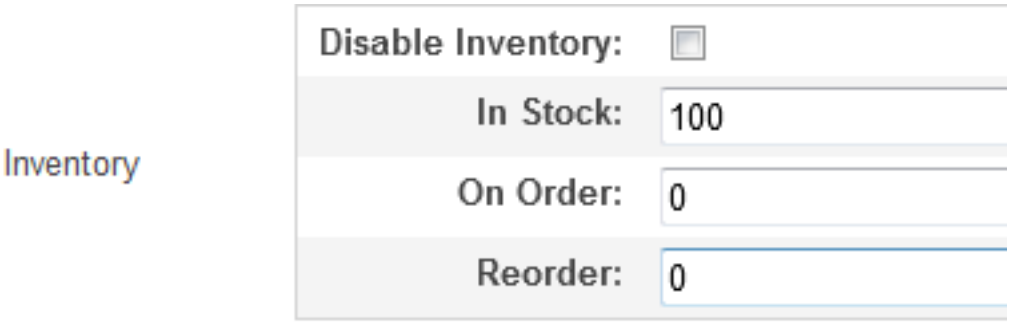


By default, this button appears when the following conditions are met:

- The product is buyable. That is, there is a check mark in the **Buyable** property for a product in the Workarea.



- The product has an **In Stock** quantity of at least one.



- The `TemplateCart` property has a cart's template location defined.

In addition, if a product has an alias path associated with it, you need to:

- Make sure the `TemplateCart` property's path is relative to the site root. For example:

```
TemplateCart="Developer/Commerce/CartDemo.aspx"
```

- Add the following to the code-behind page. This example is in C#:

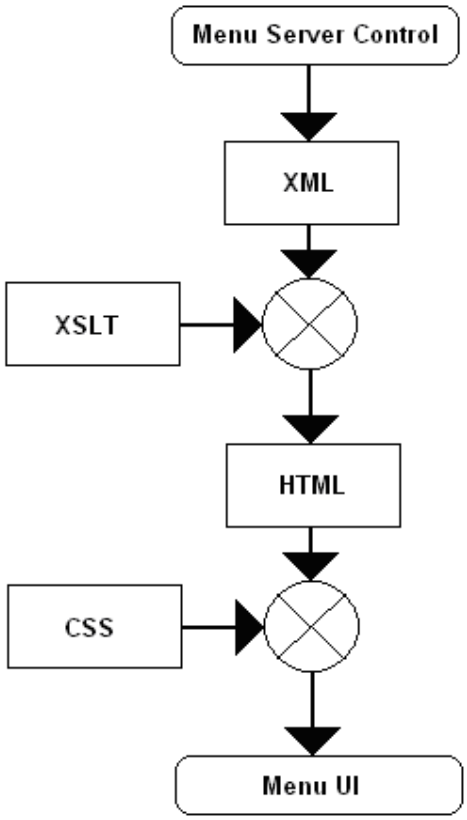
```
protected void Page_Init(object sender, EventArgs e)
{ Utilities.RegisterBaseUrl(this.Page); }
```

FlexMenu

8.50 and higher

IMPORTANT: Starting from release 8.6, the FlexMenu server control was replaced by the FrameworkUI: <ektron:MenuView> templated server control. If you are already using the FlexMenu server control, you can continue to do so, but Ektron recommends using current versions of functionality.

The FlexMenu server control displays a menu on a Web form. A FlexMenu creates XML, so you can modify its behavior using an XSLT file, and change its appearance using a cascading style sheet (.css) file.



Before you can use this server control, you must create one or more menus in the Workarea. See also: *Menu* on page 2493, Contrasting Menu server controls.

This section also contains the following topics.

Inserting the FlewMenu server control onto a page

PREREQUISITE

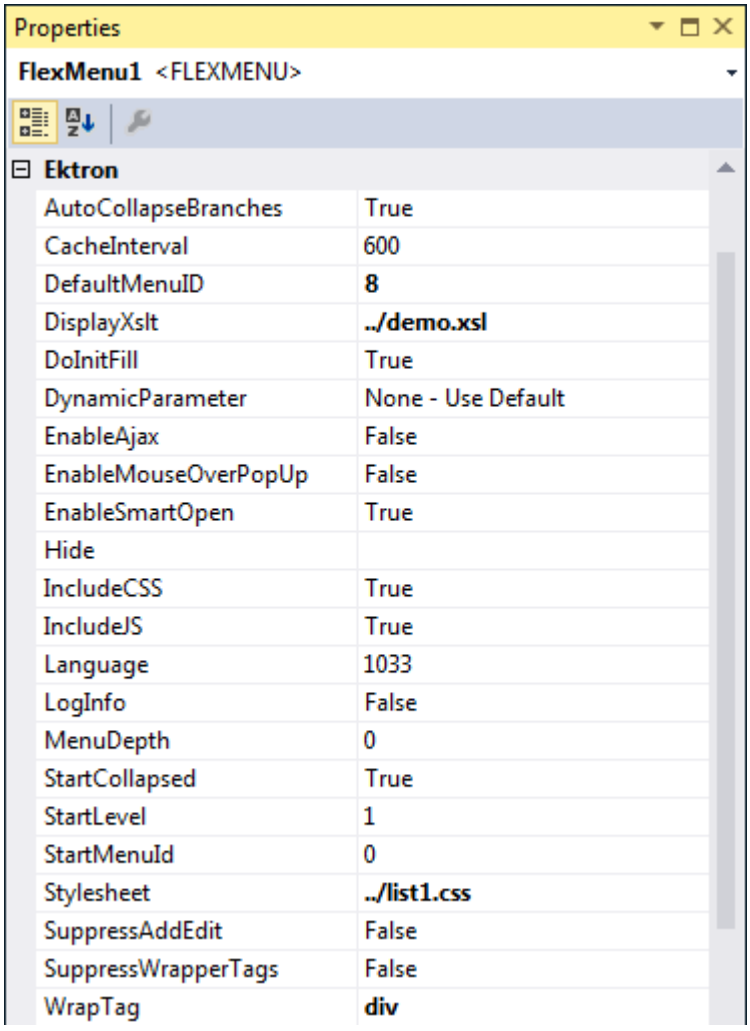
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **FlewMenu** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:FlewMenu ID="FlewMenu1" runat="server" />
```

4. Click on `FlewMenu` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



FlexMenu properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AutoCollapseBranches** (Boolean)
  - **True**. Whenever a new submenu opens, all other submenus close.
  - **False**. Other submenus remain open when a new one opens.
- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **DefaultMenuID** (Long)

  The ID of a menu that appears where you insert this server control if no other menu is identified or available. If you don't know the ID number of the menu, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147

  NOTE: If you want to change the Default menu's Starting menu, use the StartMenuID property.
- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page. If nothing is specified, the menu is output as raw XML. FlexMenus use an .xsl file to control the menu's behavior, and a .css file to control its display. Ektron provides several sample menus, and each has an xslt file. If this is a new menu, you may find it easier to copy and edit an xslt file provided with a sample menu. See also: *SampleMenu* on page 2496.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.
- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a FlexMenu ID dynamically. Set to **None. Use Default** if you want to always display the default menu.
  - **None. Use Default**. Use the default FlexMenu ID list.
  - **ekfrm**. Reads a form block's ID dynamically.

  - You may display FlexMenus dynamically by entering any value other than **id**. **id** is the default parameter for **PostParameter**.
- **EnableAjax** (Boolean)

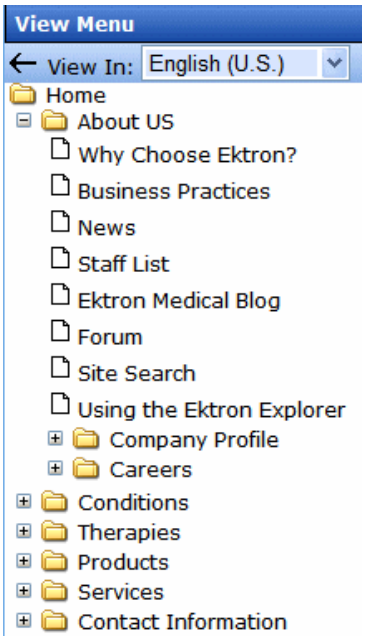
  Set to **True** to enable Ajax, which only downloads sub-menus as needed.
- **EnableMouseOverPopup** (Boolean)
  - **True**. Submenus appear as soon as the cursor moves over them.
  - **False**. Submenus only appear if a site visitor clicks them or a keyboard equivalent.
- **EnableSmartOpen** (Boolean)

  Lets you prevent submenus from opening by default. Under some circumstances, such submenus look cluttered.
  - **True**. Submenu can open automatically. To learn how to do this, see Assigning a Folder or Template to a Menu.
  - **False**. Even if all required conditions are present, submenus on a FlexMenu do not automatically open.
- **Hide** (Boolean)

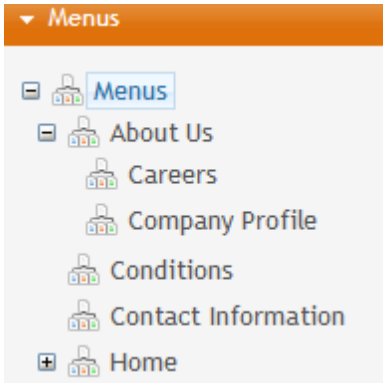
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **IncludeCSS** (Boolean)
  - **True**. Load the menu's cascading stylesheet (CSS).
  - **False**. Ignore the menu's CSS.
- **IncludeJS** (Boolean)
  - **True**. Load the menu's JavaScript.
  - **False**. Ignore the menu's JavaScript. This would be useful if you redefine the menu's output with an XSLT.
- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **MenuDepth** (Integer)

  To let site visitors browse through all menu levels, enter zero (0). To restrict site visitors to a menu level, enter the number of the lowest level. In the following example, if you set this property to **1**, a site visitor can browse through the **About Us** menu options but would not see the level 2 options (**Company Profile** and **Careers**).



- **StartCollapsed** (Boolean)
  If you set to **True**, when the menu first appears, all submenus are closed.
- **StartLevel** (Integer)
  Enter a number to indicate the level at which you want this menu to display when it first appears. To begin the menu display at the root level, enter zero (0). In the following example, the **Home** folder is level 0. The others are level 1.



  A site visitor can click a menu option to navigate to folders below the displayed level.

  NOTE: If you set a StartMenuID, the StartLevel property is ignored.

- **StartMenuId** (Integer)
  Use this property to have the default menu begin somewhere other than its root. That is, at any submenu under the default menu. For example, menu ID 46 (Products) is the default menu. However when that menu displays, you only want to show its submenus, beginning with Products Support (ID 58). In this case, enter **46** into the `DefaultMenuID` property and **58** into the `StartMenuID` property.

NOTE: If you set a `StartMenuID`, the StartLevel property is ignored.

- **Stylesheet** (String)

  Enter the style sheet that will determine the appearance of the menus. FlexMenus use an .xsl file to control their behavior, and a .css file to control their display. Ektron provides several sample menus, and each has a .css file. If this is a new menu, you may find it easier to copy and edit a .css file provided with a sample menu. See also: *SampleMenu* on page 2496.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **SuppressAddEdit** (Boolean)

  When set to **True**, suppress the Add and Edit buttons on the menu when a user is logged in to Ektron. The default is False.

  - **True**. Suppress the Add and Edit button when a user is logged in to Ektron.
  - **False**. Show the Add and Edit buttons when a user is logged in to Ektron.

- **SuppressWrapperTags** (Boolean)

  This property is set to `false` because Ajax uses `<div>` tags to rewrite the region around the tag. You *cannot* change the value to `true`.

- **WrapTag** (String)

  Lets a developer specify a server control's tag.

  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Working with the FlexMenu Xslt file

This section explains some non-intuitive elements of the *.xslt file.

- MenuFragment

  ```
  <xsl:when test="/MenuDataResult/Info/menuFragment='false'">
  ```

  A flag that indicates the XML data is not complete. Instead, it's a fragment that begins deeper than the top level (for example, a submenu fragment).

  Because the data is incomplete, the XSLT processes the fragment differently. For example, don't generate JavaScript startup code.

- menuConst

  ```
  <xsl:attributename="id"><xsl:value-ofselect="$menuConst"/>0_
  ekflexmenu</xsl:attribute>
  ```

Each menu generates several elements, which the client code (JavaScript) accesses via a unique ID. For example, JavaScript needs to identify the selected submenu or item when a user clicks on an element.

`menuConst` is only used is when creating elements without a corresponding XML block, such as when creating a structure to hold the menu.

- #NoScroll

  ```
  <xsl:attribute name="href">#NoScroll </xsl:attribute>
  ```

  `NoScroll` is sent to the `href` portion of a link when there is nothing to put there (for example, when the link is supposed to run JavaScript).

  `NoScroll` prevents the page from refreshing, going to another page, or scrolling when it should not.

- event

  ```
  <xsl:attribute name="onkeydown">returnekFlexMenu.menuBtnKeyHdlr
  (event);</xsl:attribute>
  ```

  To make the xslt cross-browser compatible, it must support different methods of obtaining/passing the event object. In this example, the global event object is passed to the handling function.

  NOTE: `event` corresponds to `window.event`. `window;` is implied.

How FlexMenu determines which item is selected

The FlexMenu server control can select (highlight) options in a FlexMenu as site visitors navigate around the website. For example, a site visitor arrives at a Web page through a link in an email. If the QueryString matches the an item in a FlexMenu, the item is shown as selected.

Below is the logic the FlexMenu uses to decide when a menu item should show as selected. The list is presented in the order in which the code checks to see if an item should be highlighted.

If any menu item is selected, its parent menu is marked selected. If any menu is selected, all ancestor menus are selected. When all tests have been performed and no matches are found, the FlexMenu is rendered with no items selected.

NOTE: As soon as one condition is satisfied, the item is shown as selected and the testing stops.

NOTE: Steps 2 through 8 are each repeated recursively throughout the menu data hierarchy until either a match is found or the end is reached. If there are no matches for a test, the control continues with the next one.

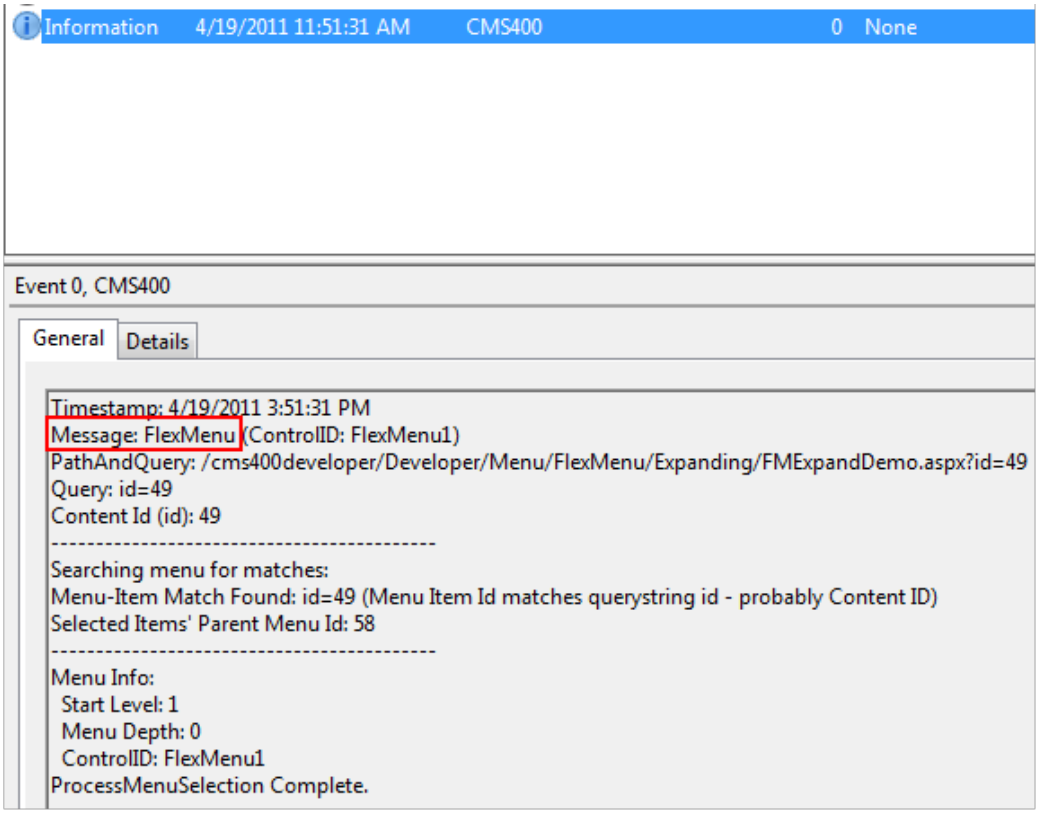
1. Inspects the QueryString to see if the `ekxmensel` parameter is present with a matching menu node ID. This parameter is used to specify the exact node a user clicked. The node, its parent, and ancestor menus are all marked as selected.
2. Inspects the QueryString to see if a Content ID parameter value matches a menu item. If the Content ID in the QueryString matches a menu item, it is

highlighted. For example, the QueryString includes `ExamplePage.aspx?id=123`, and a menu item links to Content ID 123.

3. Inspects the QueryString to see if a Form ID parameter value matches a menu item. If the Form ID passed in the QueryString matches a FlexMenu item, it is highlighted. For example, the QueryString includes `ExamplePage.aspx?ekfrm=456`, and a menu item links to Form ID of 456.
4. Inspects the QueryString for a direct match with a menu item link. If there is a match, the menu item is selected. For example, if the QueryString has `ExamplePage.aspx` and there is an item on the menu that matches, the menu item is shown as selected.
5. Inspects the QueryString for a `id`, `ekfrm` or `pageid` parameter. If one is found, the control looks for a folder association between submenus and the folder that contains the object with the given ID. If the association exists, the menu item is shown as selected.
6. Inspects the QueryString to see if there is a direct match with a menu button link. If there is a match, the menu button is selected. For example, the QueryString has `ExamplePage.aspx` and there is a button on the menu that matches.
7. Inspects the QueryString to see if there's a template association with the filename. If there's a match, the menu item is shown as selected.
8. Inspects the QueryString for the `id` or `ekfrm` parameter and whether a value greater than zero is associated with it. If so, the control checks menu buttons for use of `LinkIt.aspx`. If a button is using `LinkIt.aspx` and either `id` or `ekfrm` parameter matches, the menu button is selected.

Analyzing FlexMenu's selection of menu items

The FlexMenu server control performs a complex sequence of tasks to determine which menu option to select. If you are unsure of a FlexMenu's behavior, you can view a log that lists the reasons for a menu selection. Use the log to troubleshoot issues with a FlexMenu.



Setting up log of FlexMenu information

Follow these steps to set up the ability to view a log of FlexMenu display information.

1. Open your *site root*`/web.config` file.
2. Under `system.diagnostics`, set `LogLevel` to **4**. Save and close web.config.

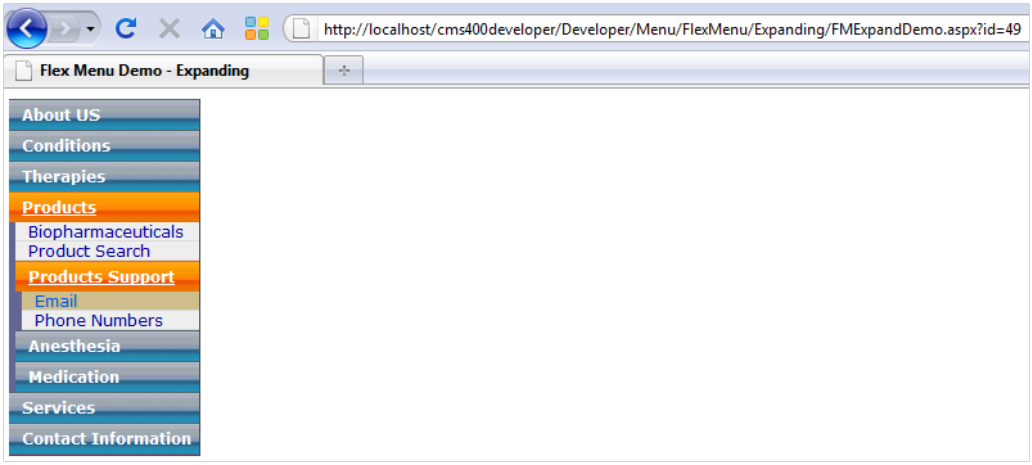
```
<system.diagnostics>
  <switches>
    <!-- Determines the level of messages that are logged
      1 = Error:  Only Errors are logged.
      2 = Warning:  Only warnings and Errors are logged.
      3 = Information:  Only Informationals, Warnings, and Er
      4 = Verbose:  Everything is logged.

      NOTE: you can configure where each message level is log
    -->
    <add name="LogLevel" value="4" />
  </switches>
</system.diagnostics>
<ektronCommerce>
  <add key="ek_ecom_ComplianceMode" value="false" />
  <!-- This is used only when compliance mode is on, and canno
  <add key="ek_ecom_PasswordHistory" value="4" />
```

3. Edit the .aspx or .ascx page that hosts the FlexMenu control.
4. Set the control's `LogInfo` property to **true** and save the page.

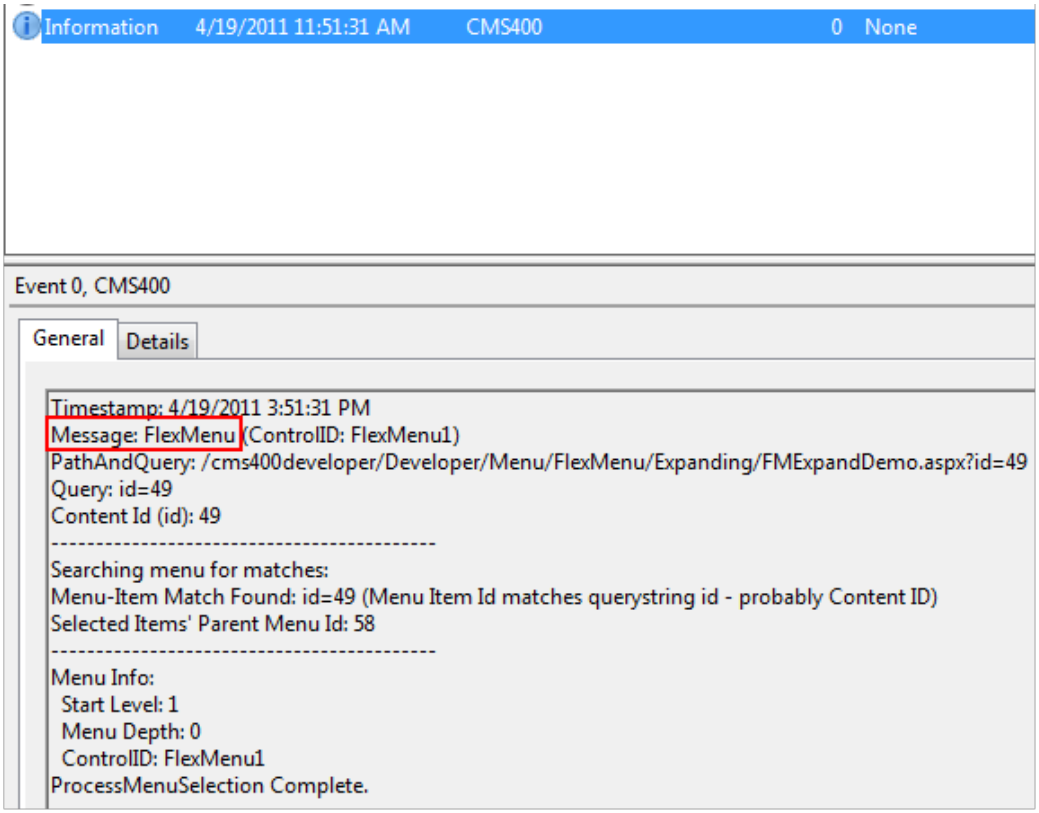
```
<CMS:FlexMenu ID="FlexMenu1" LogInfo="True" runat="server" />
```

NOTE: It may be helpful to temporarily disable the menu's cache by setting `CacheInterval="0"`.

5. Refresh the browser that contains the FlexMenu you edited in the Step 4.



6. Using the Windows Event Viewer, look for the newest information-level event that contains **Message: FlexMenu** (under the timestamp line).

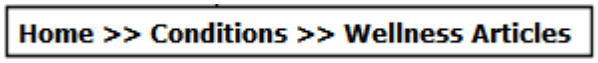


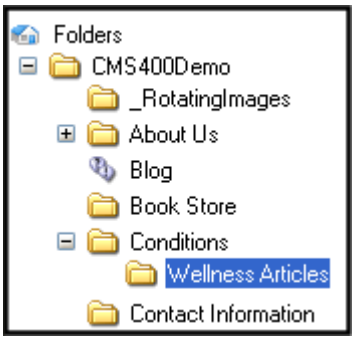
The event message identifies the reason the menu node was selected. Or, if no matches were found, the message indicates that nothing is selected.

FolderBreadcrumb

8.50 and higher

The FolderBreadcrumb server control displays Sitemap breadcrumbs. This trail typically consists of the current content's folder path. For example, the FolderBreadcrumb below matches the content's folder structure.





The FolderBreadcrumb server control does not read your folder structure and display its path. Instead, administrators define a folder's breadcrumb trail on the Folder properties > **Breadcrumb** tab.

Inserting the FolderBreadcrumb server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

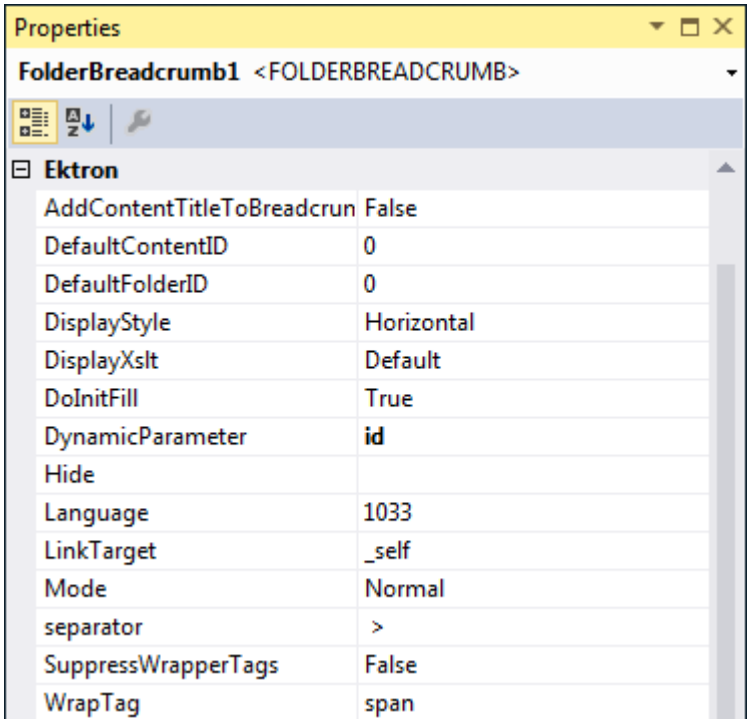
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **FolderBreadcrumb** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:FolderBreadcrumb ID="FolderBreadcrumb1" runat="server" />
   ```

4. Click on `FolderBreadcrumb` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as

you modify the property values.



FolderBreadCrumb properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AddContentTitleToBreadcrumb** (Boolean)

  When set to **true**, the content title is added to the end of the breadcrumb path, as shown.

  Home > Content > Content_Title

  By default, the value is set to **false**. In that case, the breadcrumb path looks like this.

  Home > Content

- **DefaultContentID** (Long)

  Display the breadcrumb trail for the folder in which the given content ID resides. To use this property, breadcrumb information for a folder must be defined in the Workarea > [Specified Folder] > Folder properties > Breadcrumb tab.

- **DefaultFolderID** (Long)

  The folder ID for which you want the breadcrumb trail to display. If a DefaultContentID is given, it overrides this property. To use this property, breadcrumb information for a folder must be defined in the Workarea > [Specified Folder] > Folder properties > Breadcrumb tab.

- **DisplayStyle** (DisplayStyles)

  Indicate how to display the breadcrumb trail: horizontally or vertically. The default is Horizontal.

- **DisplayXslt** (String)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page. If nothing is specified, the menu is output as raw XML. FlexMenus use an .xsl file to control the menu's behavior, and a .css file to control its display. Ektron provides several sample menus, and each has an xslt file. If this is a new menu, you may find it easier to copy and edit an xslt file provided with a sample menu. See also: *SampleMenu* on page 2496.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a content ID dynamically.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LinkTarget** (ItemLinkTargets)

  Determines the type of window that appears when you click a link in the server control.

  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.

- **Mode** (Modes)

  Lets you make the breadcrumb trail appear as non-hyperlinked plain text.

  - **Normal** (normal). Breadcrumb trail is hyperlinked
  - **DisplayOnly**. Breadcrumb trail is plain text

- **Separator** (String)
  Enter one or more characters to separate the items in a breadcrumb trail on this Web form. The default character is the greater than sign (**>**).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Using a FolderBreadcrumb server control

Add the FolderBreadcrumb server control to each Web form for which you want to create a breadcrumb site map.

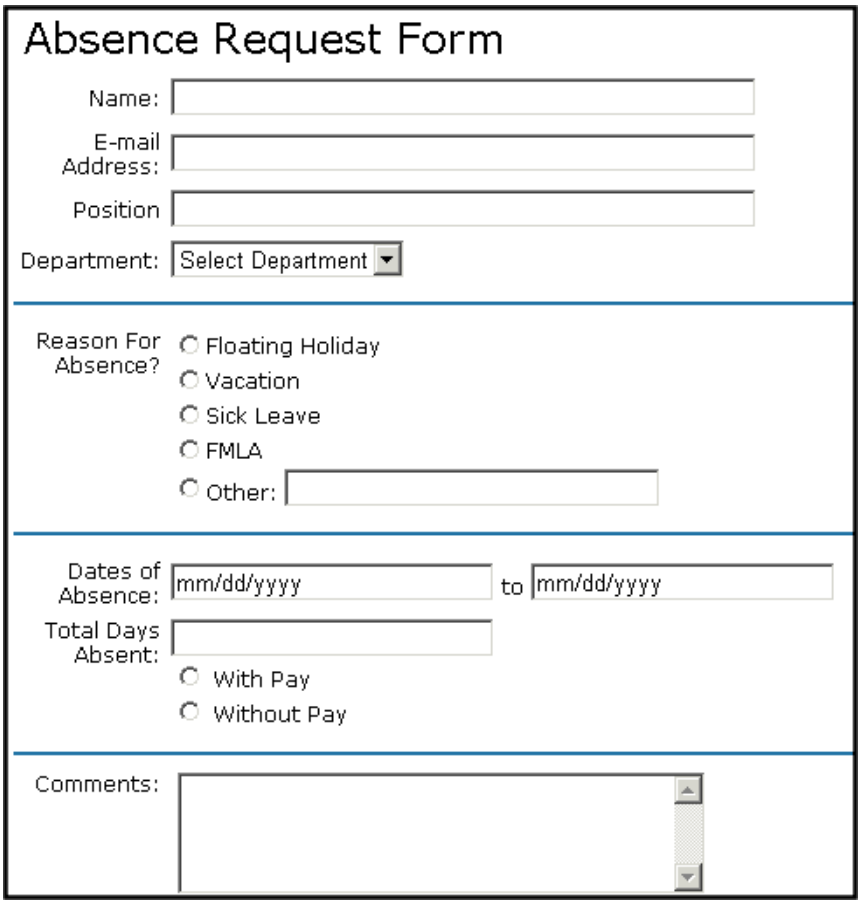
Follow these steps to use a FolderBreadcrumb server control.

1. In the Workarea, define breadcrumb information for a folder. See also: *BreadCrumb* on page 2196
2. Open a Web form for which you want to create a site map.
3. Drag and drop the FolderBreadcrumb server control onto the Web form.
4. Set the `DefaultContentID` or `DefaultFolderID` property. If you use `DefaultContentID`, make sure the content is in a folder for which breadcrumb information is defined. If you use `DefaultFolderID`, make sure breadcrumb information is defined for that folder.
5. Save the Web form.
6. Open a browser.
7. View a Web page that hosts the control to view the breadcrumb trail.

FormBlock

8.50 and higher

The FormBlock server control displays a content block associated with a form. When added to a template and visited, the form content block might look like the following example. You can change the display to suit your needs by modifying its properties. See also: Working with HTML Forms.



IMPORTANT: If you create a template for an existing form content block, you must manually change its quicklink to point to the new template. This change does not occur automatically. This procedure is described in Adding a Quicklink to Content.

Inserting the FormBlock server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

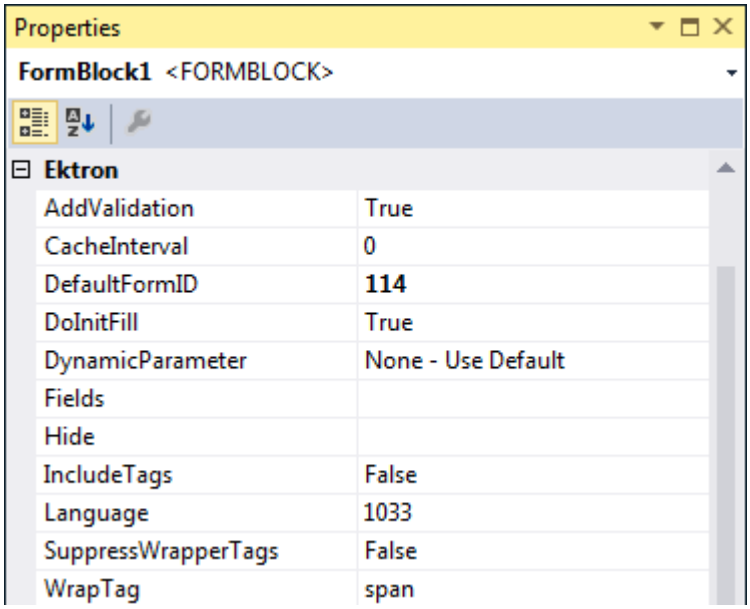
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.

3. Drag the **FormBlock** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:FormBlock ID="FormBlock1" runat="server" />
   ```

4. Click on `FormBlock` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



FormBlock properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AddValidation** (Boolean)

  The AddValidation property is obsolete and ignored. It has no effect. It is always true.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultFormID** (Long)

  The ID of a FormBlock that appears where you inserted this server control if no other form block is identified. If you don't know the ID number of the form block, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147.

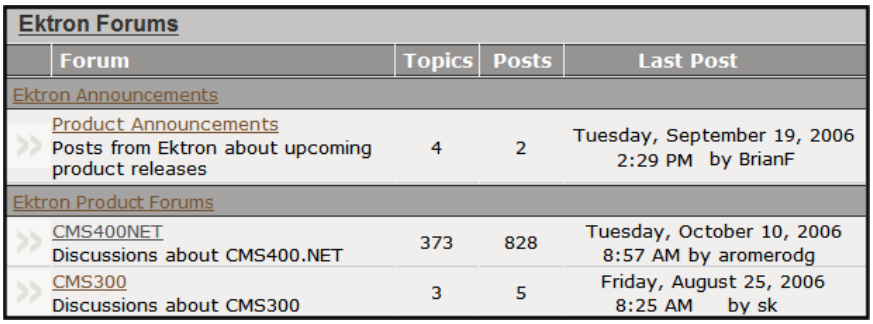
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)
  To make this form block dynamic, select **id**. When you do, this server control uses the form block passed as a URL parameter.
- **Fields** (FormFieldCollection)
  Displays a list of fields that are defined in the form. These fields are read only. This is an excellent way of displaying the field names used on the form. With this list of names, you can create events using the fields without having to enter the Workarea to see the names.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **IncludeTags** (Boolean)
  Determines if tags are generated automatically of manually.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Forum

8.50 and higher

After creating at least one hierarchy of discussion board elements, place a Forum server control on a Web page. The Forum server control displays a discussion board on a Web page. You should add text below the Login server control to remind the site

visitor to enter an email address at the **User** field. For example, "At the **User** field, enter your email address." See also: Working with Discussion Boards.



Ektron Forums

Forum	Topics	Posts	Last Post
Ektron Announcements			
Product Announcements Posts from Ektron about upcoming product releases	4	2	Tuesday, September 19, 2006 2:29 PM by BrianF
Ektron Product Forums			
CMS400NET Discussions about CMS400.NET	373	828	Tuesday, October 10, 2006 8:57 AM by aromerodg
CMS300 Discussions about CMS300	3	5	Friday, August 25, 2006 8:25 AM by sk

If you want to require site visitors to authenticate, the Web page that hosts the forum should contain

- a Membership server control (or a link to page that has one). This lets site visitors/membership users register for discussion boards.
- a Login server control that lets the site visitor/membership user log in

IMPORTANT: After placing a Forum server control on a page, follow the procedure described in *Updating the page Command* on page 2443. Otherwise, the user may get an error when posting a reply.

Inserting the Forum server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

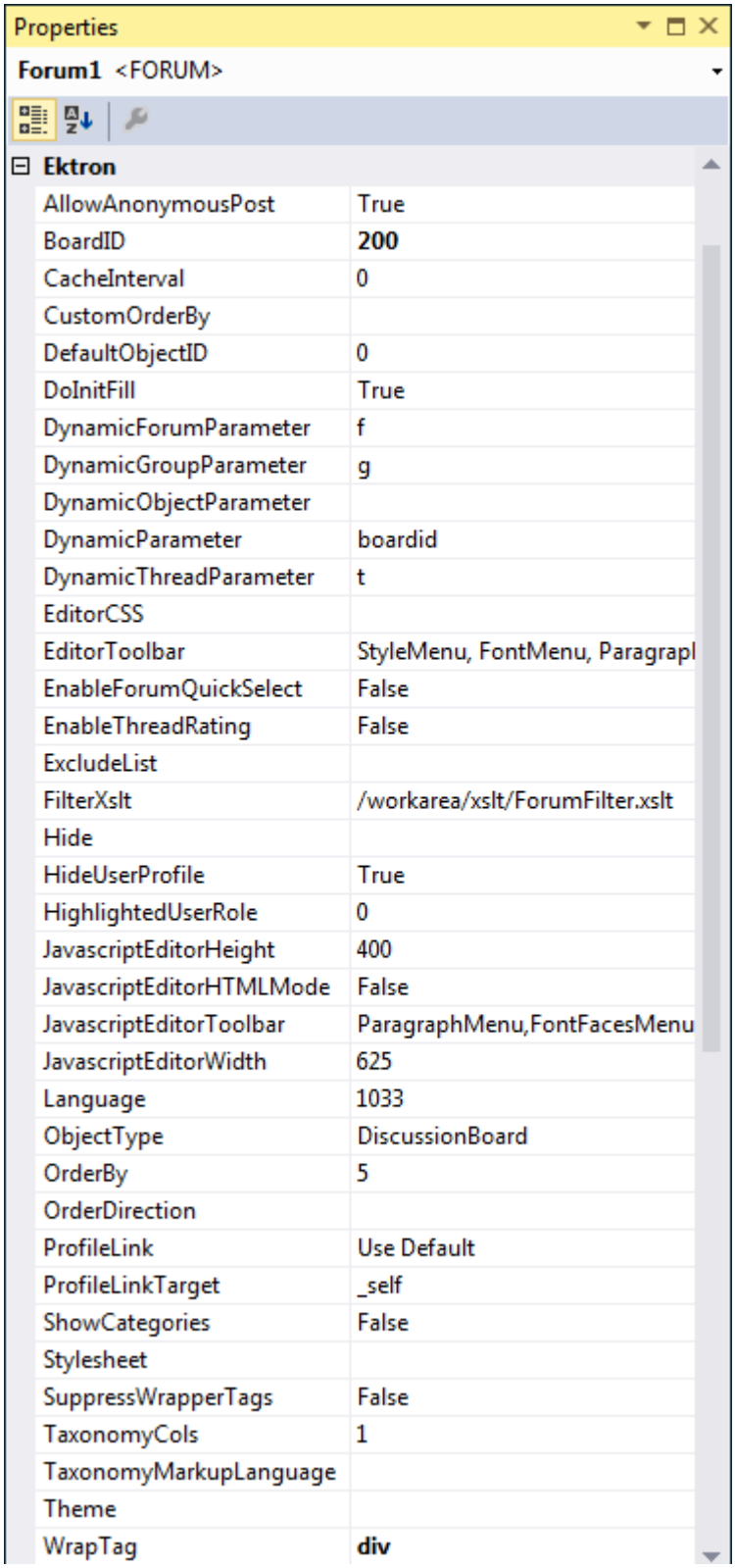
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Forum** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:Forum ID="Forum1" runat="server" />
   ```

4. Click on `Forum` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the

property values.

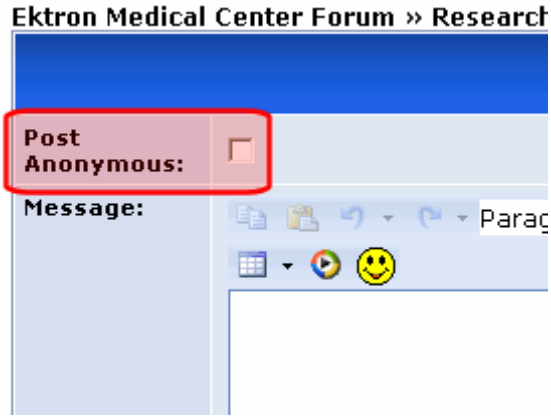


Forum properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AllowAnonymousPost** (Boolean)

  Set to True to allow site visitors the option of posting anonymously to the forum. When true, a **Post Anonymous** checkbox appears above the text editor used to create a post.



  If a site visitor adds a check mark to this box, the word Anonymous appears where the Display Name normally appears. The default is **True**.

  - **True**. Display **Post Anonymous** checkbox when site visitors create a post.
  - **False**. Disable **Post Anonymous** checkbox.

- **BoardID** (Long)

  The ID of the discussion board to display on this page if one is not defined in a query string parameter. If you don't know the ID, click **Ellipses** (), then sign in and select a discussion board.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **CustomOrderBy** (String)

  Provide a property's Friendly Name to order search results by that property. For example, if you define DocAuthor, results will be sorted by the document's author. Results can be ascending or descending based on `OrderDirection`. If you enter an invalid property, no search results are returned. If you specify both `CustomOrderBy` and `OrderBy`, the `OrderBy` property is ignored.

- **DefaultObjectID** (Long)

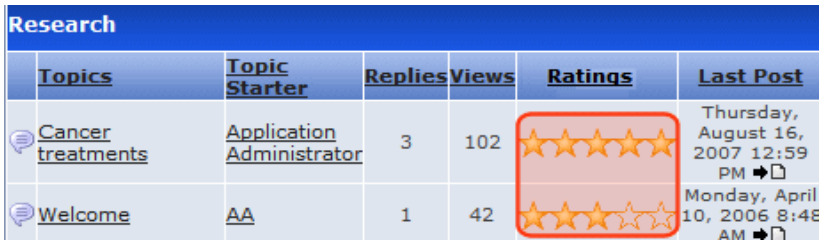
  The Static ID of a community group.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during

the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicForumParameter** (String)
  Gets or sets the QueryString parameter to read a forum ID dynamically. The default is "**f**". Note that a Forum resides one level below a discussion board.
- **DynamicGroupParameter** (String)
  Gets or sets the QueryString parameter to read a group's ID dynamically. The default is "**g**".
- **DynamicObjectParameter** (String)
  Dynamic Parameter for the community group id. Default is "**id**".
- **DynamicParameter** (String)
  Gets or sets the QueryString parameter to read a discussion board's ID dynamically. Set to **None - Use Default**. If you want to always display the default discussion board. Note that a discussion board is one level above a Forum.
  - **boardid**. Reads a Threaded discussion board's ID dynamically.
  - **None - Use Default**. Use the default discussion board's ID.
  - **ID**. Reads a discussion board's ID dynamically.
- **DynamicThreadParameter** (String)
  Gets or sets the QueryString parameter to read an individual thread's ID dynamically. The default is "**t**". Note that a Thread resides 2 levels below a discussion board and one level below a Forum.
- **EditorCSS** (String)
  Set the style sheet for the Editor when a site visitor creates or edits a post. By default, this property is blank. When a style sheet is not supplied, the style sheet defined in the Theme property is used.
- **EditorToolbar** (String)
  Enter a comma separated list of items that you want to appear on the Editor's toolbar. The default is **StyleMenu, FontMenu, ParagraphMenu, TextFormatMenu, LinkMenu, ClipBoardMenu, SymbolsMenu, EmoticonSelect, WMV, Table**. See also: *Adding and removing toolbar items from the editor* on page 2444.
- **EnableForumQuickSelect** (Boolean)
  If you set this property to `true`, a drop-down list of all forums appears below the topic list. The site visitor can click a forum and jump immediately to it.
- **EnableThreadRating** (Boolean)
  If you set this property to `true`, a ContentReview server control appears on any topic screen run by this server control. The site visitor can use the control to rate the topic thread. An average rating for the thread appears next to each topic on the forum screen.



- **ExcludeList** (String)

  Enter a comma-separated list of custom user properties to exclude from the Forum's profile page. For example, to suppress the **Subscriptions** field value, enter **subscriptions**.

- **FilterXslt** (String)

  Enter the path to an XSLT file used to filter forum content, such as, HTML Attributes, Tags and unwanted words in a user's forum post. The path can be relative or absolute. The user's post is filtered when the user clicks **Submit**.

  By default, this property points to `<webroot>/siteroot/Workarea/Xslt/ForumFilter.xslt`. This file removes hrefs with javascript:, vbscript: and "on" events in the link. You can modify this file or create a new one.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **HideUserProfile** (Boolean)
  - **True**. Hide user profiles from non-administrator users.
  - **False**. Display user profiles for non-administrator users.

  The default is **True**. An administrator can view a profile regardless of how this property is set.

  On the other hand, if the **Private Profile** is set to Private for any user, the profile information is not visible, regardless of this setting. If Private Profile is set to Colleagues, only colleagues can see profile information. When the profile is visible, only properties and their values not listed in the `ExcludeList` property (above) appear.

- **JavascriptEditorHeight** (Integer)

  Set the height in pixels for the eWebEdit400 editor. The default is **400**. The minimum height is 300.

- **JavascriptEditorWidth** (Integer)
  Set the width in pixels for the eWebEdit400 editor. The default is **625**. The minimum width is 500.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

  Set property to **-1** if you want the Forum server control to display topics from all available languages.
- **ObjectType** (String)
  Describes the type of forum displayed by this server control.
  - **CommunityGroup**. Used for forums inside a Community page.
  - **DiscussionBoard**. Used for forums on templates that are not Community pages.
- **OrderBy** (String)
  The order of search results. For example, you want to sort search results by last modified date.
  - **Title**. Content title (alphabetical).
  - **ID**. Content ID number.
  - **Date Created**. Date the content was created.
  - **Date Modified**. Date the content was most recently modified.
  - **Editor.** User who last edited the content (alphabetical).
  - **Rank**. Rank assigned to the content.

  NOTE: The **OrderDirection** field determines the *direction* of the search results. For example, if you sort by ID and **OrderDirection** is set to **Descending**, the results sort by ContentID number with the highest number at the top of the list.

  IMPORTANT: Specifying a `CustomOrderBy` property overrides this property.
- **OrderDirection** (String)
  The direction in which search results are sorted. The default is **Ascending**.
  - **Ascending**. Alphabetical results from A to Z; numeric values low to high; dates from oldest to most recent
  - **Descending**. Alphabetical results from Z to A; numeric values high to low; dates from most recent to oldest
- **ProfileLink** (ItemLinkTargets)
  Enter a link to the user's social networking profile page, a part of Ektron's community platform. This allows a user to click another user's name link or avatar and be taken to the user's profile page. The link has 2 variables that represent the user's ID and display name.
  - **{0}**. User ID
  - **{1}**. User display name

You need to have both variables in the link. The Web form can be relative or absolute. For example:

```
userprofilepage.aspx?uid={0}&dn={1}
```

The default for this property is `?g=profileu={0}`.

When the default for this property is used, users are forwarded to a user's profile page that is included with the forum control.

- **ProfileLinkTarget**

  Determines the type of window that appears when you click a link in the server control.

  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **ShowCategories** (Boolean)

  If set to **True**, when this server control appears, the user see a **Filter by Category** option. This option helps a site visitor zero in on relevant content. If **False**, the **Filter by Category** option does not appear.
- **Stylesheet** (String)

  Enter the style sheet that will determine the appearance of the menus. FlexMenus use an .xsl file to control their behavior, and a .css file to control their display. Ektron provides several sample menus, and each has a .css file. If this is a new menu, you may find it easier to copy and edit a .css file provided with a sample menu. See also: *SampleMenu* on page 2496.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.
- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TaxonomyCols** (String)

  Use this property to determine the number of columns on the Taxonomy screen available from this discussion board. In the illustration below, taxonomy categories are arranged in 3 columns (the default value).

- **TaxonomyMarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.See also: *Ektron Markup Language* on page 2633
- **Theme** (String)
  Enter the custom theme's folder name. The theme can be relative to the site root or located in the current folder. For example:
  - Relative. `Theme="/workarea/csslib/themes/winter"`
  - Current Folder. `Theme="mytheme"`

  If you do not specify a theme, the property uses the location defined by the discussion board's CSS theme property in the Workarea. See also: *Using a custom theme* on the next page
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Updating the page Command

To prevent an error from appearing when a user posts a reply to the Web page that hosts the Forum server control:

1. Open the Web form onto which you inserted the Forum server control.
2. Access Source view.
3. Go to the top line of the Web form, which begins with `@ Page`.

```
<%@ PageLanguage="VB"AutoEventWireup="false"
CodeFile="Default2.aspx.vb"Inherits="Default2" %>
```

4. Near the end of the line, enter `validaterequest=false`.
5. Build the page.

Using a custom theme

This property lets you specify a custom location for your themes. By doing so, you prevent them from being overwritten when you upgrade.

If you do not specify a theme in the server control, Ektron uses what is defined in the discussion board's properties **CSS Theme** field.

To create a custom theme:

1. Create a new subfolder on your site. In this example, we use a winter theme:

```
<web root>/<your site>/Workarea/csslib/winter.
```

2. Copy all files in the `/Workarea/Threadeddisc/themes` folder to the `winter` folder.
3. Change the image files to match your theme. To find the name of an image:
   a. Right click on it while viewing it in a browser.
   b. Select Properties. The name appears at the top of the dialog box.
4. Change the discussion board's .css file to match your theme and save it with a custom name. (You do not need to do this if an existing .css file meets your theme's needs.)

   IMPORTANT: Make sure the **CSS theme** property in the Workarea is pointing to the proper CSS file.

5. View the discussion board in a browser to see the changes.

Adding and removing toolbar items from the editor

IMPORTANT: You cannot create new buttons and add them to the Forum Editor. You can only add and remove existing buttons.

IMPORTANT: This section is only applicable if you are using the eWebEdit400 editor.

You can add and remove toolbar items on the Forum Editor by editing the EditorToolbar property. This property contains a series of string values that represent each item, listed below.

- **StyleMenu**. Display a list of paragraph styles. Users can select from the list to apply a style to selected text.
- **FontMenu**. Display a list of available font styles, sizes and colors.

- **ParagraphMenu**. Controls the display of buttons that affect a paragraph. This includes:
  - Numbered List
  - Bullet List
  - Outdent
  - Indent
  - Align Left
  - Align Center
  - Align Right
  - Justify
  - Horizontal Rule
- **TextFormatMenu**. Controls the display of buttons that format text. This includes:
  - Bold
  - Italics
  - Underline
- **LinkMenu**. Controls the display of buttons that allow a user to add and remove hyperlinks links and work with the library. This includes:
  - Hyperlink Manager
  - Remove Link
  - Library
- **ClipBoardMenu**. Controls clipboard buttons that allow a user to cut, copy, and paste content. This item also controls the Undo and Redo buttons.
- **SymbolsMenu**. Controls the display of the Symbol button, which allows users to insert symbols and special characters.
- **EmoticonSelect**. Controls the display of the Emoticon button.
- **WMV**. Controls the display of the Insert WMV button.
- **Table**. Controls the display of the Insert table button.

ImageControl

8.50 and higher

The ImageControl server control, when viewed on a Web form, displays an image stored within the Ektron Document Management feature. If you log in and have permission to edit the image, you can right click the image and select **Edit**. This action displays a dialog box with which the user can update the image.

IMPORTANT: Images are stored as assets using the Document Management feature. This control does not use the Ektron Library.

Inserting the ImageControl server control onto a page

PREREQUISITE

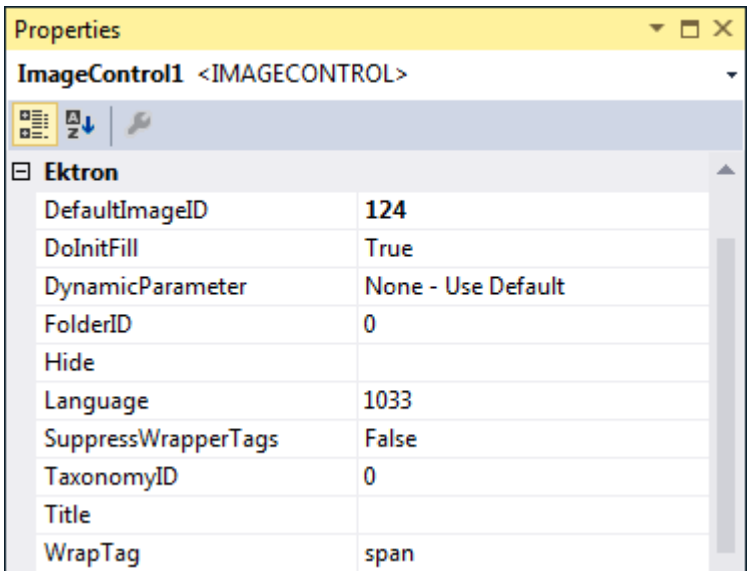
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **ImageControl** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:ImageControl ID="ImageControl1" runat="server" />
   ```

4. Click on `ImageControl` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



ImageControl properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **DefaultImageID** (Long)

  The image asset's content ID you want to display. If you don't know the ID number of the asset, use the CMS Explorer to browse to it.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)
  Gets or sets the QueryString parameter to read an image asset's ID dynamically. To have the default image ID used, leave blank.
- **FolderID** (Long)
  The ID of the folder where images are added. If you don't know the ID number of the folder, use the CMS Explorer to browse to it.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TaxonomyID** (long)
  The ID of the taxonomy to which assets are added.
- **Title** (String)
  Set the image's alt/title text. By default, the image file name is used.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Language server controls

8.50 and higher

The Language server controls are as follows:

- *LanguageAPI* below
- *LanguageSelect* on page 2450

LanguageAPI

8.50 and higher

The LanguageAPI server control lets a developer force a particular language for a website. You can do this by dropping the server control on the page and setting a language in the `SiteLanguage` property box. You can also override site language logic by programmatically using the LanguageAPI server control to detect the browser's language, and display the site in that language.

Inserting the LanguageAPI server control onto a page

PREREQUISITE

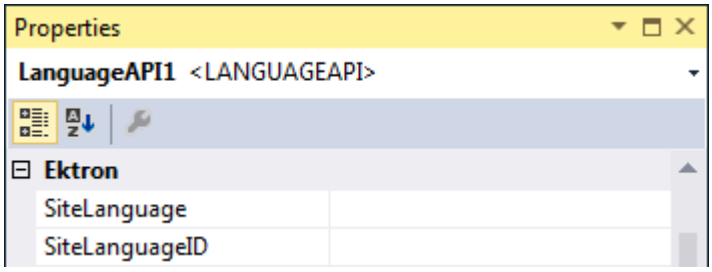
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **LanguageAPI** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:LanguageAPI ID="LanguageAPI1" runat="server" />
   ```

4. Click on `LanguageAPI` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



LanguageAPI properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **SiteLanguage** (String)

  Sets the site language. Runs through the IsValid function to verify that the language is an active language in the system.

- **SiteLanguageID** (Integer)

  Numeric value of the site language. This is the property you will use if you are using code-behind to set the sites language ID.

  Example: 1036 = French

For a list of supported languages, go to **Workarea > Settings > Configuration > Language Settings**.

LanguageAPI code-behind Only properties and Methods

The following is a list can only be used programmatically.

- **CurrentLanguage** (String)
  Read only. Returns the current language.
- **CurrentLanguageID** (Integer)
  Read only. Returns the current language ID.
- **DefaultLanguage** (String)
  Read only. What the default language of the site is set to. For example, the demo site is "English (Standard)".
- **DefaultLanguageID** (Integer)
  Read only. Returns the value that is the default language ID of the site. For example, the demo site is "1033" for English.
- **GetLanguage** (Integer Argument)
  This method returns a string. Pass in a valid language ID and it will return the language name.
- **GetLanguageID** (Sting Argument)
  This method returns an integer. Pass in a valid language name and it will return the language ID.
- **IsValid** (Argument)
  This method returns a boolean. You can pass in a language ID or a string and it will tell you if the system is supporting it.
- **LanguageIdList** (Array of Integers)
  Read only. Lists all the language IDs that are activated in Ektron. For information on how to enable languages, see Determining Which Languages are Available.
- **LanguageTitleList** (Array of Strings)
  Read only. Lists all the languages that are activated in Ektron.
- **MultiLanguageEnabled** (Boolean)
  Read only. Tells if the site supports multi-language mode.
  - **True** (default). Multi-language enabled
  - **False**. Multi-language not enabled

Using LanguageAPI programmatically

This example uses a logo that is not managed through Ektron. It retrieves the current language from the LanguageAPI control, and uses that information to choose the logo version to display. The code-behind looks like this.

```
Select Case LanguageAPI1.CurrentLanguageID
       Case 1031
       Image1.ImageUrl = "germanlogo.png"
       Case 1033
       Image1.ImageUrl = "englishlogo.png"
       Case 1036
       Image1.ImageUrl = "frenchlogo.png"
End Select
```

LanguageSelect

8.50 and higher

The LanguageSelect server control displays a language selection drop-down list on an Ektron Web page. The control lets a site visitor select a language in which to view the site. Here is what the control looks like when published on a Web page.

It lists all languages selected on the **Settings > Configuration > Language settings** screen. (For more information, see Determining Which Languages are Available.)

You can place this control in any location of any page on your site.

Inserting the LanguageSelect server control onto a page

PREREQUISITE

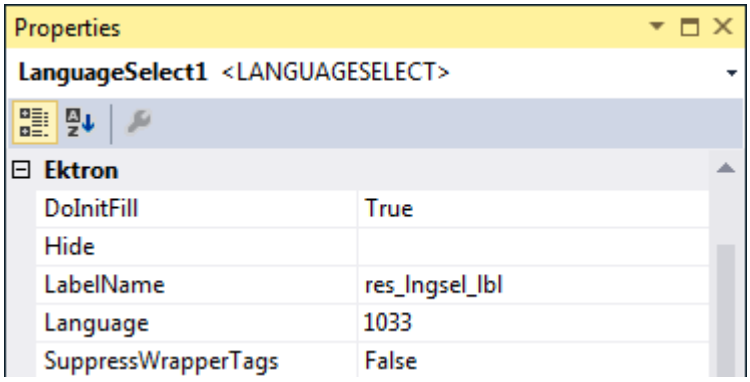
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **LanguageSelect** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:LanguageSelect ID="LanguageSelect1" runat="server" />
```

4. Click on `LanguageSelect` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



LanguageSelect properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

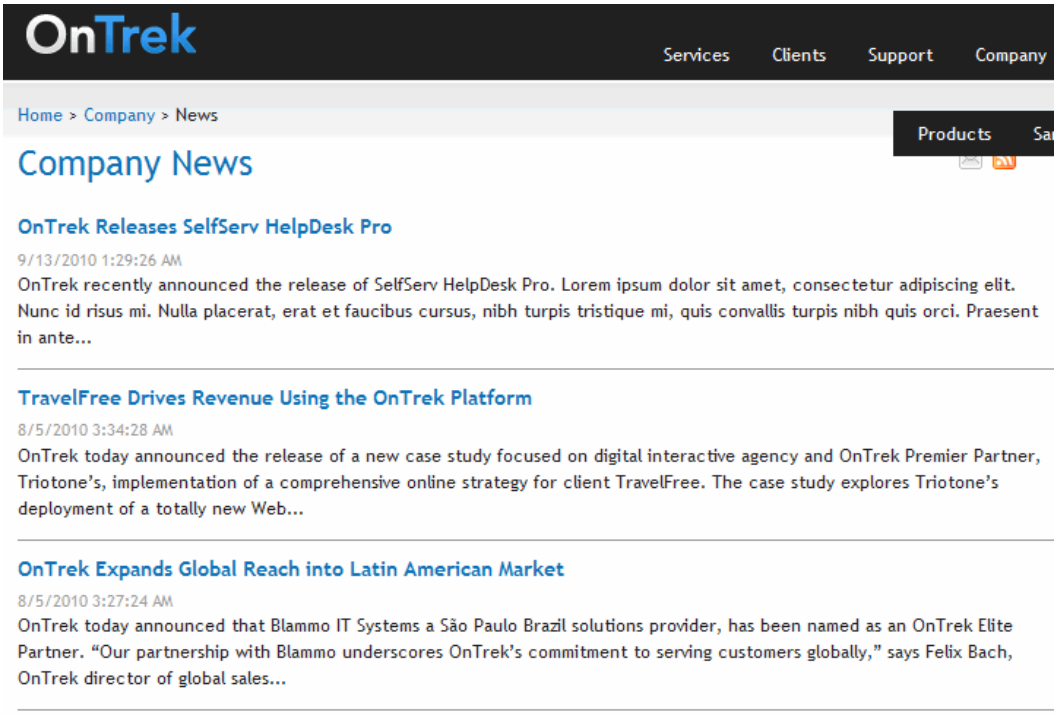
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **LabelName** (String)
  Lets you define the label next to the language select drop-down box.
- **Language** (Integer)
  Set a language for the Language Select Box. This property shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

ListSummary

8.50 and higher

IMPORTANT: Starting from release 8.60, the ListSummary server control was replaced by the FrameworkUI: <ektron:ContentView> templated server control. If you are already using the ListSummary server control, you can continue to do so, but Ektron recommends using current versions of functionality.

The ListSummary server control displays a list of content in a selected folder on a Web page. Optionally, the display can include content in all subfolders of the selected folder. When added to a template and visited, a ListSummary looks like this.



It displays each content block's title and summary information. You can modify the display to suit your needs by modifying its properties.

NOTE: To display a List Summary on a PageBuilder page, use the ListSummary widget.

This section also contains the following topics.

NOTE: In contrast to a List Summary, a ContentList server control displays selected content items from any Ektron folder. See also: *ContentList* on page 2336

Inserting the ListSummary server control onto a page

PREREQUISITE

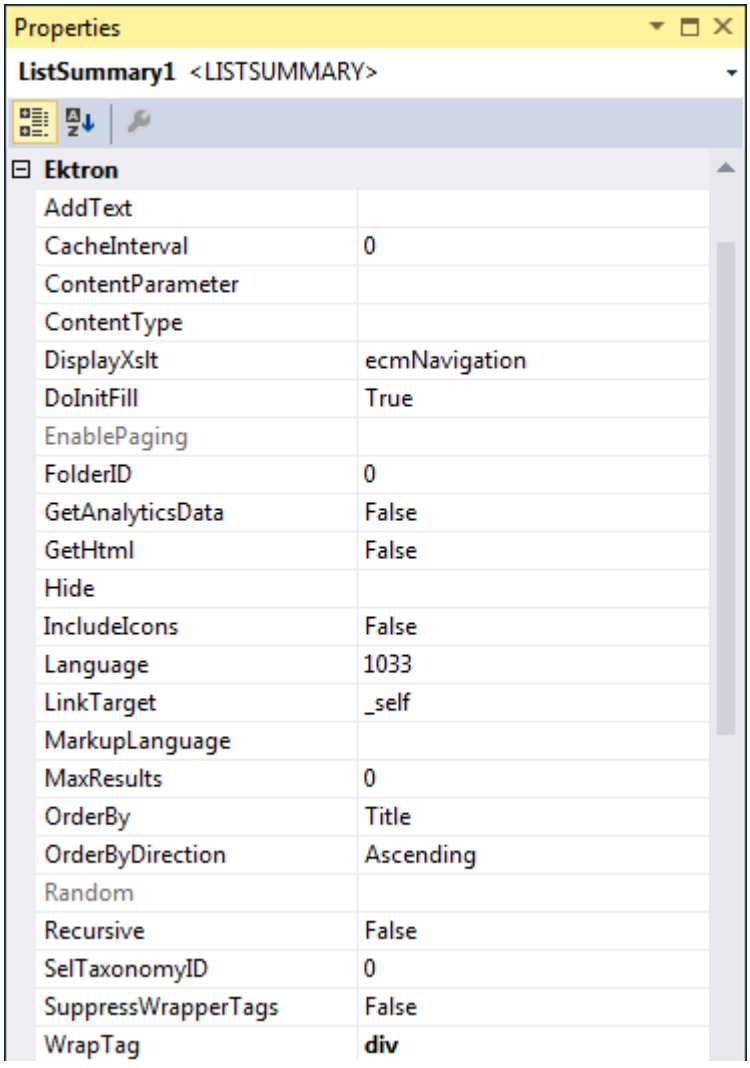
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **ListSummary** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:ListSummary ID="ListSummary1" runat="server" />
```

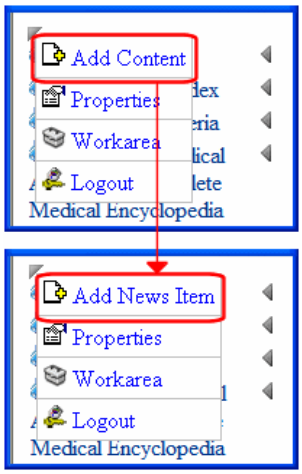
4. Click on `ListSummary` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



ListSummary properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AddText** (String)
  Override the control's default text for the Add Content menu item.
  For example, you have a News website. You could change **Add Content** to **Add News Item**.



- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

  NOTE: If the `EnablePaging` property is set to True, the `CacheInterval` property is disabled.

- **ContentParameter** (String)
  Checks the QueryString for this value and replaces the list summary with a content block when specified. Leave blank to always display the list summary.

- **ContentType** (String)
  Determines the type of content that appears in the list summary. The default is Content. Choices are:

  - **AllTypes**. All content types for the given folder
  - **Content**. A list of content items.
  - **Forms**. Forms appear in the list summary
  - **Archive_Content**. Archived content blocks appear in the list summary
  - **Archive_Forms**. Archived forms appear in the list summary
  - **Assets**. Assets, such as offices documents, appear in the list summary
  - **Archive_Assets**. Archived assets appear in the list summary
  - **LibraryItem**. Library items appear in the list summary
  - **Multimedia**. Multimedia items appear in the list summary
  - **Archive_Media**. Archived multimedia items appear in the list summary
  - **NonLibraryContent**. All content types except library items.
  - **DiscussionTopic**. Forum topics appear in the list summary.

- **CatalogEntry**. Displays all catalog entries (products) for a specified catalog.

- **DisplayXslt** (String)
  Determines how information on the page appears
  - **None**. Databind only
  - **ecmNavigation**. Lists the title of every content block in the folder.
  - **ecmTeaser**. Lists the title of every content block in the folder plus the content summary.
  - **Path to Custom Xslt**. If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

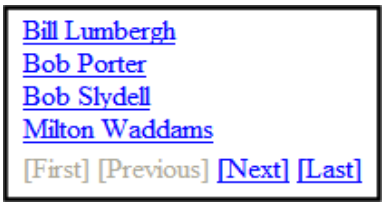
WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

NOTE: If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.

- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **EnablePaging** (Boolean)
  This property, in conjunction with the **MaxNumber** property, lets site visitors view an unlimited number of items while controlling the amount of screen space. The **MaxNumber** property limits the number of items displayed. If you set this property to **True**, and the number of items exceeds **MaxNumber**, navigation aids appear below the last item, allowing the visitor to go to the next screen.

  See example below.



  So, for example, if a List Summary has 9 items and the `MaxResults` property is set to 3, the screen displays only the first 3 items. When the site visitor clicks **[Next]**, he sees items 4, 5 and 6, and so on.
  - **True**. Use paging feature
  - **False**. Ignore paging feature

NOTE: If the `EnablePaging` property is set to True, the `CacheInterval` property is disabled.

- **FolderID** (Long)

  The folder that contains the items which appear in the list summary. See also: *Browsing your Ektron site using CMS Explorer* on page 2147. The `Recursive` property determines whether content blocks in this folder's child folders also appear.

- **GetAnalyticsData** (Boolean)

  Set this property to **True** if you want the following information for each content in the list. Returns **Content View Count**, **Content Rating**, **Content Rating Average**. Create your own XSLT styles to display this data.

  IMPORTANT: This property provides reliable data only when the Business Analytics Feature is on. Analyzing Websites.

- **GetHtml** (Boolean)

  Set to **True** to display the HTML body for all content in the list summary. For example, to display content inside a Web server control such as a GridView.

  - **True**. Get and display HTML for each content block in the list summary
  - **False**. Do not get and display HTML.

- **Hide** (Boolean)

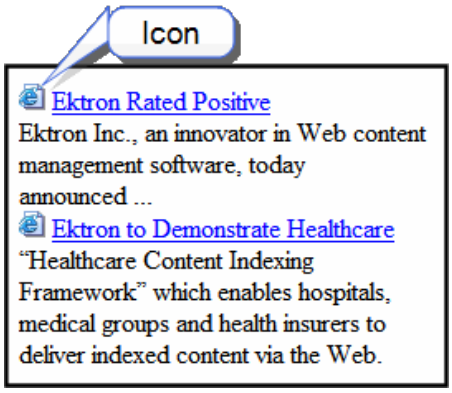
  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **IncludeIcons** (Boolean)

  Choose whether to display icons next to the list summary's links.

  IMPORTANT: This property works only when ecmSummary or ecmTeaser are used in the `DisplayXslt` property. When the `[$ImageIcon]` variable is used in an EkML file and that file is assigned to the `MarkupLanguage` property, this property acts as True. See also: *Ektron Markup Language* on page 2633.



- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **LinkTarget** (String)
  Determines the type of window that appears when you click a link in the server control.
  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **MarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.

  The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.

  To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.

  See also: *Ektron Markup Language* on page 2633

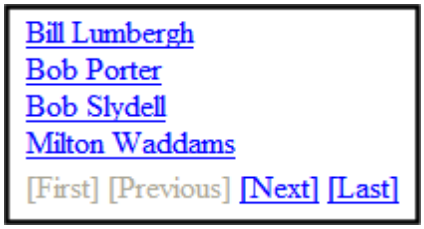
  If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored. If the EkML file contains the `[$ImageIcon]` variable, the `IncludeIcons` property acts as True.

  See also: *listsummary.ekml* on page 2635
- **MaxResults** (Integer)
  Enter the maximum number of items to appear in the initial display of this server control.

  If you enter no value or 0 (zero), the maximum is 50. This is done for performance reasons.

  To let site visitors view more than the maximum but limit the amount of space being occupied, enter the maximum number of results per page here. Then, set the `EnablePaging` property to **True**.

  If you do and more than the number of `MaxResults` are available, navigation aids appear below the last item to help the site visitor view additional items. See example below.



- **OrderBy** (Ektron.Cms.Controls.CmsWebService.TeasersOrderBy)

  NOTE: For releases 8.0.1 and later the **OrderKey** property has been replaced by this **OrderBy** property.

  Sort the list by one of the values.

  - **Title**. Content Title
  - **DateModified**. Date content last modified
  - **DateCreated**. Date content created
  - **LastEditorFname**. First name of user who last edited content
  - **LastEditorLname**. Last name of user who last edited content
  - **Start Date**. Go Live date of content
  - **Rated**. Business Analytics Content Rating
  - **ContentViewCount**. Business Analytics Content Views
- **OrderbyDirection** (Ektron.Cms.Controls.CmsWebService.OrderByDirection)

  How to order the hyperlinks on the list. The sort field is determined by the `OrderKey` property.

  - **ascending**. Hyperlinks are arranged A, B, C or 1,2,3.
  - **descending**. Hyperlinks are arranged Z, Y, X or 3,2,1

  If sorting by date, descending puts the most recent first.
- **Random** (Boolean)

  Set to **True** if you want to randomly display one content block in the specified folder. The content changes each time a user views the page.

  - **True**. Randomly display one content block.
  - **False**. Display the list summary normally.

  If you use a custom XSLT or EkML file, the type of content displayed can be manipulated. For example, if you use an EkML file that has the [$Html] variable in it, the actual content appears instead of a link. See also: *Ektron Markup Language* on page 2633 and *[$Html]* on page 2666
- **Recursive** (Boolean)

  Determines if the display includes content in child folders of the selected folder.

  - **True**. Include content from child folders.
  - **False**. Do not include content from child folders.

- **SelTaxonomyID** (Integer)
  Set the ID of the taxonomy that content will be associated with when a logged in site visitor uses the Silver Access Point's **Add HTML Content** to add content to a list summary server control.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Retrieving the XML structure of a list summary

Retrieving the XML structure of XML content allows for greater control over developing XSLs. The following example shows how to retrieve the XML structure.

1. Open a new Web form.
2. Drag and drop a ListSummary server control onto it.
3. Set the `FolderID` property.
4. Drag and drop a textbox on the Web form.
5. Set the `TextMode` property to MultiLine.

   NOTE: You should set the width of the text box to at least 400px.

6. On the code-behind page, add the following line.

```
Textbox1.Text = ListSummary1.XmlDoc.InnerXml
```

7. Build the project.
8. View the Web form in a browser.
9. The ListSummary's XML structure appears in the textbox.

Login

8.50 and higher

The Login server control places a login button on the template when displayed in a browser. The Login server control displays the following buttons on a Web page. See also: Managing Logins and Passwords.

. When user is not logged in, this button appears. Clicking the button opens the login window, where a user can enter a username and password. Upon authentication, the user is logged in to the Ektron website.

. After a user logs in, this button replaces the login button to let the user log out.

. When logged in, this button appears under the logout button, allowing the user to access the Workarea.

. Lets the user preview the entire website as if all checked-in content were published.

. Turns off site preview mode.

. Launches Ektron online help.

Placing a Login button

You can add any number of login buttons to a template. You can insert a login button on each template, or set up a special Web page, called login.aspx, from which users can log into the Ektron site without the public being able to access the page.

Inserting the Login server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

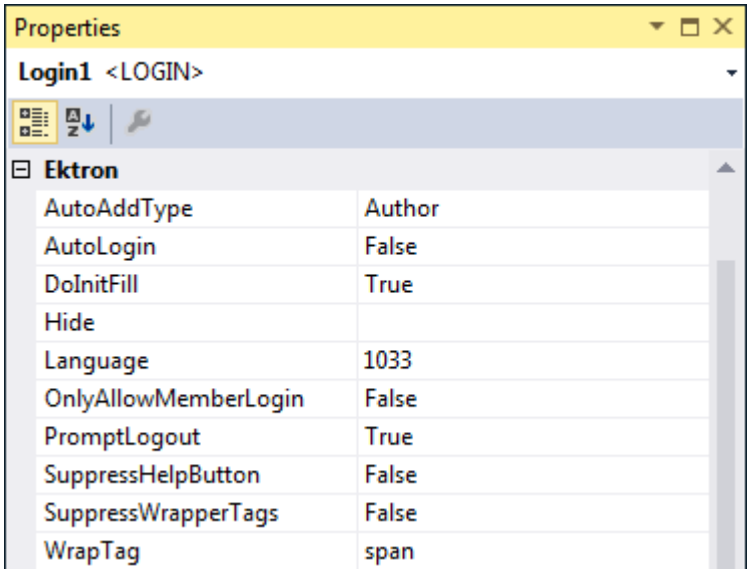
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Login** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Login ID="Login1" runat="server" />
```

4. Click on `Login` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the

property values.



Login properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AutoAddType** (Common.EkEnumeration.AutoAddUserType)

  When using Single Signon, the Login server control can be used to add users to Ektron. In this scenario, when a user signs on with Active Directory credentials, that user is created within the Ektron database. Use this property to define the type of user that is automatically added to Ektron. See also: Single Sign On.

  - **Author.** Ektron user
  - **Member**. Membership user

- **AutoLogin** (Boolean)

  If this property is set to true and Active Directory Integration is enabled, users are automatically logged in using Active Directory authentication. They do not need to enter a username or password. See also: Single Sign On.

  - **True**. Use Active Directory authentication when logging in.
  - **False** (default). Do not use Active Directory authentication when logging in.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

- **True**. Hide the control output.
- **False**. Display the control output.

- **Language** (Integer)

  Set a language for the Login server control. This property shows results in design-time (in Visual Studio) and at run-time (in a browser).

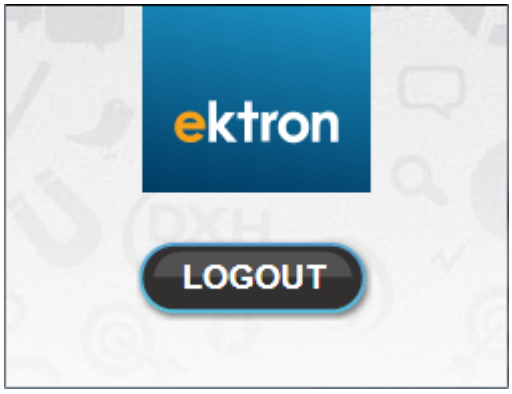
- **OnlyAllowMemberLogin** (Boolean)

  Allows only membership users to log in. This property prevent users from logging as an Ektron user and accessing the Workarea. If an Ektron user tries to log in using this control, this message appears: "Only members are allowed to login here." The default is **False**.

  - **True**. Only membership users can log in
  - **False**. Ektron users and membership users can log in

- **PromptLogout** (Boolean)

  When set to False, the logout process omits the Logout window.

  - **True**. Users must click **Logout** to log out.
  - **False**. Logout window does not appear.

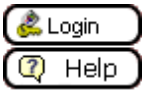
- **SuppressHelpButton** (Boolean)

  Hides or displays the Help button. When displayed, the button appears below the Login button.

  - **True**. Do not display the Help button.

  - **False** (default). Display Help button.

  If you are editing this server control from a text file and want to suppress the Help button, add the following code to the login tag source:

  ```
  <CMS:Login ID="Login1" runat="server" SuppressHelpButton="True" />
  ```

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

- **True**. Suppress wrap tags.
- **False** (default). Allow wrap tags.

- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Map

8.50 and higher

The Map server control displays a map that flags locations of interest. Each location is an Ektron content item to which map information was added. For example, if your site hosts a school district, each location could represent one school.

Inserting the Map server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

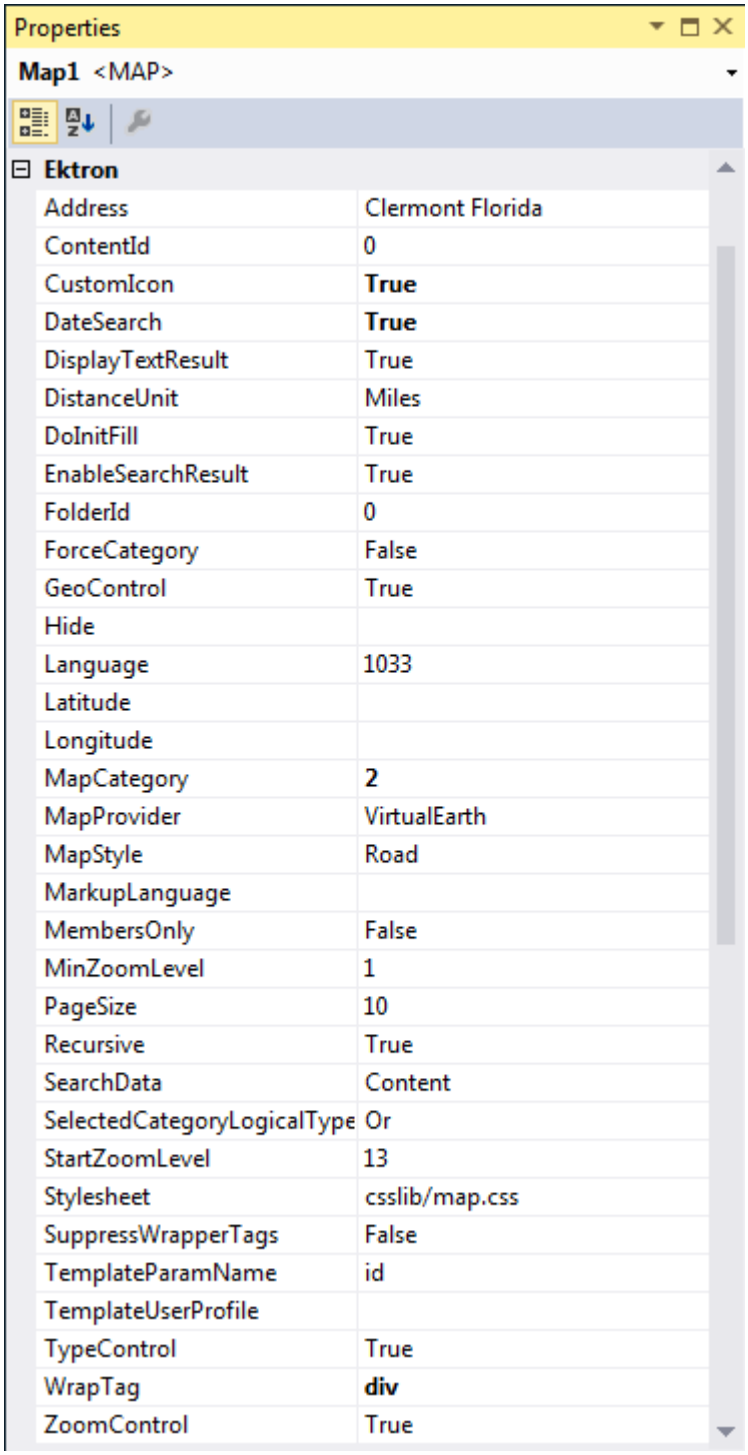
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Map** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Map ID="Map1" runat="server" />
```

4. Click on `Map` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the

property values.



Map properties

Within Visual Studio, you cannot see the map in design mode. However, you can right mouse click the mouse and select **View in Browser** to see the effect of changing the properties.

You cannot place more than one map server control on a form.

Troubleshooting

- If Map does not render on Web page, change the VirtualEarthMap `web.config` element as follows:

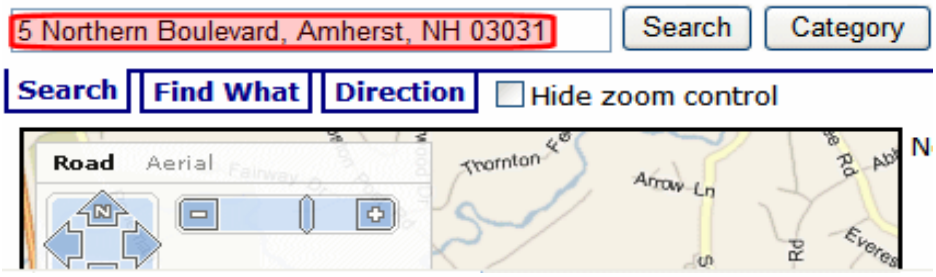
```
<add key="VirtualEarthMap"
  value="https://dev.virtualearth.net/mapcontrol/
  mapcontrol.ashx?v=6.1&amp;s=1" />
```

- If a user must click a warning away before users can see map on page, upgrade Google Maps to a premier subscription. For more information, see Google Maps for Business.

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **Address** (String)

  To set a default map center, enter its address. The address appears in the **Search** field when the map first displays. If you only enter a zip code, the map centers on its post office. If you enter an address, the `latitude` and `longitude` properties are ignored.

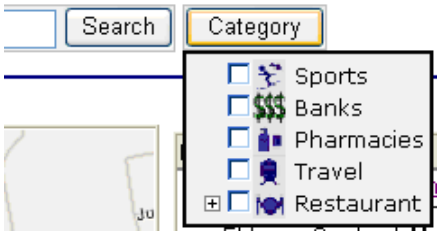


- **ContentId** (Long)

  Use this field in conjunction with the **EnableSearchResult** field to limit the map to a single content item. Here, you identify the content item to be mapped. Content *must* have latitude and longitude values to appear on a map.

- **CustomIcon** (Boolean)

  Use this field if you want the Category popup box to display an icon to the left of each category, as shown below.

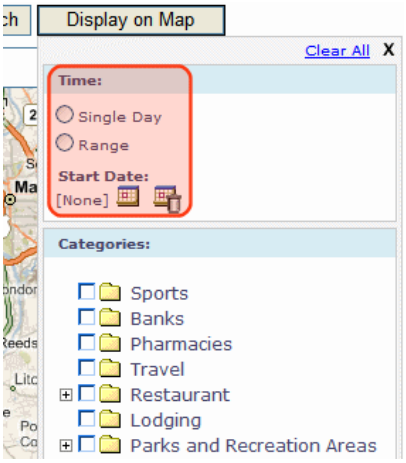


  To use this value, open the `webroot/Workarea/images/application/maps/tree` folder. In that folder, create a new folder whose name is the same as the Taxonomy category assigned to the Map server control at the `MapCategory` property. (In the sample

site, this Taxonomy's name is `MapCategory`.) Then, place the icons in that folder by category title name with a .png extension. Use an underscore (_) to separate taxonomy levels. For example, the image for the category Restaurant > American must be named `Restaurant_American.png`.

- **DateSearch** (Boolean)

  Set to **True** to assign a date to content. This feature is helpful for date-related content, such as concerts, meetings, or sporting events. You can filter what appears on the map by date. If you set this property to true, assign a date to date-related content using the **MapDate** standard metadata field. Then, publish the content. When you click a map's **Display on Map** button, the popup screen includes date criteria, as shown below. You can select map items by a single date or a range of dates.



- **DisplayTextResult** (Boolean)

  If you want to display a box of information about each map item to the right of a map, enter **True**. To suppress the text box, enter **False**.

- **DistanceUnit** (Unit)

  Enter the map's units of distance. Choices are miles and kilometers. The default value is miles.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **EnableSearchResult** (Boolean)

  Use to determine if the Map server control accesses the Ektron search to return results. **True** is the default value. You would change it to **False** if you want to display a map with only one content item, that is, a single location. Specify this item at the `ContentID` property. In this case, the map does not find other Ektron content whose longitude and latitude are within the map's boundaries. Site visitors viewing the map can still use the **Search** and **Directions** tabs to get directions to the location. For example, your site features homes for sale, and you want a map to show a featured home of the week. To do so, set this

property to **False**, and enter the content that describes that home in the **ContentID** field.

- **FolderId** (Long)

  Enter the ID number of the folder whose content is searched on this map. If the `recursive` property is true, folders below this folder are also searched.

- **ForceCategory** (Boolean)

  When set to true, this property causes the map to only show content associated with the taxonomy category defined in the `MapCategory` property. When set to false, the map shows all content within the map's boundaries. For example, if 5 content blocks appear on a map and 3 are assigned to a taxonomy, set this property to True and the `MapCategory` property to the ID of the taxonomy. When a user views the map, it displays the 3 content items associated with the taxonomy.

- **GeoControl** (Boolean)

  If you want to see a search box with tabs above a map, enter **True**. To suppress the text box, enter **False**. See also: .

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **Latitude** (Decimal)

  To set a default map center via latitude and longitude (as opposed to an address), enter the latitude here.

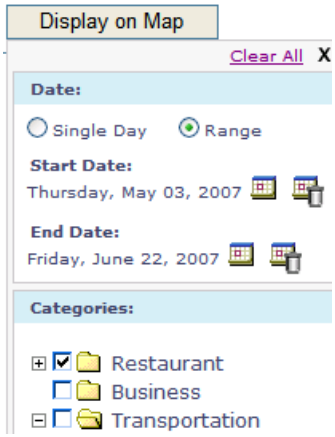
- **Longitude** (Decimal)

  To set a default map center via latitude and longitude (as opposed to an address), enter the longitude here.

- **MapCategory** (Long)

  Enter the ID number of the taxonomy whose categories appear when up click this map's **Display on Map** button. When a map first appears, all eligible content appears. If you click **Display on Map**, you can choose categories and limit the map to items assigned to them. For example, you could view restaurants only. As another example, a map could initially display all campuses in your state college system. Use the **Display on Map** pop-up window to limit the map to community colleges.

  While an OR relationship among categories is the most intuitive and common, you can set up an AND or NOT logical relationship among selected categories.



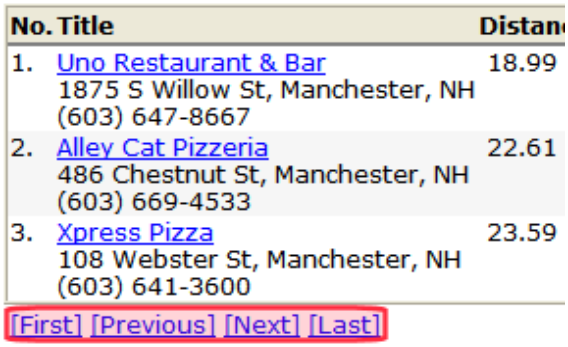
- **MapProvider** (Provider)
  Select the service that provides the map, either Google or Bing Maps for Enterprise.
- **MapStyle** (Style)
  Enter the map's display mode: Road, Satellite or Hybrid. This setting only affects Bing Maps for Enterprise maps.
- **MarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.

  The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.

  To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.

  See also: *Ektron Markup Language* on page 2633

  See also: *map.ekml* on page 2637
- **MinZoomLevel** (Integer)
  If you want to set a map zoom level below which map locations will not appear, enter that value. The default value is 4. Possible values are between 1 (most detailed) and 19.
- **PageSize** (Integer)
  Enter the number of locations that can appear on one page of the text box after a search is executed. See also: . If more than this number of locations are available, use **[First] [Previous] [Next] [Last]** at the bottom of the page to view additional locations.

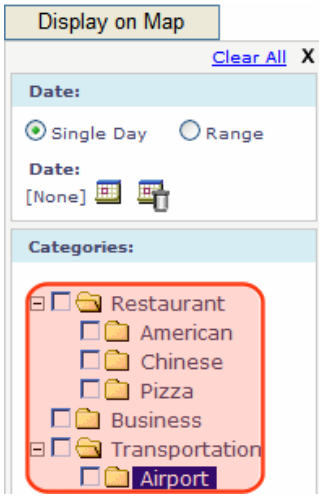


- **Recursive** (Boolean)

  In the `FolderID` property, you specify a folder whose content is searched on this map. To extend the search to all folders below this folder, set this property to **true**.

- **SelectedCategoryLogicalType** (LogicalType)

  Use this property to determine the logical relationship among several categories on the Display on Map tab. (See). There are 3 choices: OR, AND, and NOT.



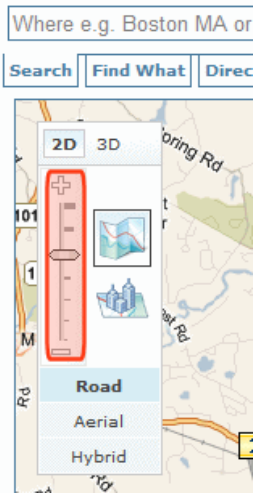
  By default, an OR relationship exists among your selections. So, for example, if all 3 Restaurant categories are checked, then any restaurant that is defined as American, Chinese or Pizza appears on the map. If you change this property's value to AND, only content to which *all* selected categories apply appear on the map. In the above example, only restaurants defined as American *and* Chinese *and* Pizza appear on the map.

  Alternatively, you can set the property's value to NOT. In this case, only content to which the selected categories are not applied appear on the map. To continue the above example, if you select **Chinese**, only restaurants that are not assigned the Chinese category appear on the map.

- **StartZoomLevel** (Integer)

  Enter the zoom level at which the map initially appears. See also: . Zoom level 1 is the least detailed, showing the entire world. Zoom level 19 is the most detailed, showing the smallest streets. By default, maps in the sample site have

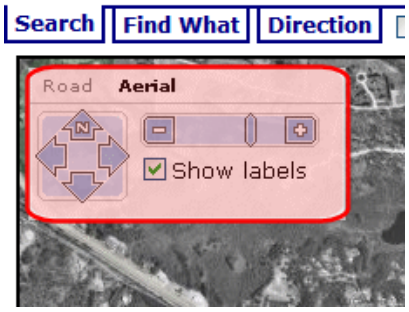
a zoom level of 12, which shows an area of about 10 miles (16 kilometers). A site visitor can adjust the level using the zoom control (circled below).



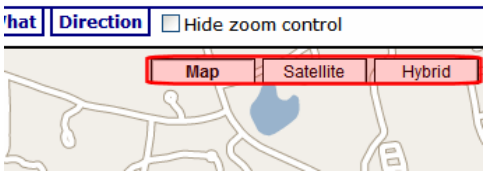
- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  NOTE: If you enter a valid EkML file at the `MarkupLanguage` property, the `Stylesheet` property is ignored.

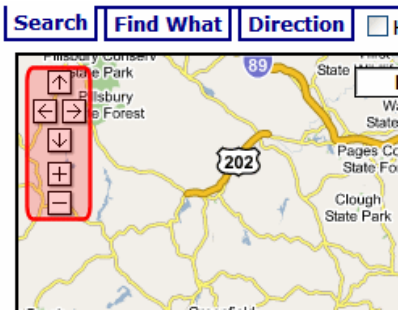
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TemplateParamName** (String)
  Sets a QueryString parameter for the ID of users that are returned in the search results. This parameter is passed to the template page defined in the `TemplateUserProfile` property when a user clicks the Location (map) icon associated with a user.
- **TemplateUserProfile** (String)
  The URL path of the user profile template. The path can be relative or absolute.
- **TypeControl** (Boolean)
  If this map uses Bing Maps for Enterprise maps, this property enables or disables the zoom/direction/type control (highlighted below). It lets you zoom the map in and out, move the center in any direction, and change the display style (Road or Aerial).



If this map uses Google Earth maps, this property enables or disables the type control (highlighted below). It lets you change the display style (Map, Satellite, or Hybrid). Use the `ZoomControl` property to display or suppress Google Earth's zoom and direction controls.



- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.
- **ZoomControl** (Boolean)
  Use this property to display or suppress the Google map zoom control (highlighted below). For Bing Maps for Enterprise maps, the `TypeControl` property determines the zoom control display.



Using the Map server control

The Map server control displays a map that flags locations of interest. Each location is an Ektron content item to which map information was added. For example, if your site hosts a school district, each location could represent one school.

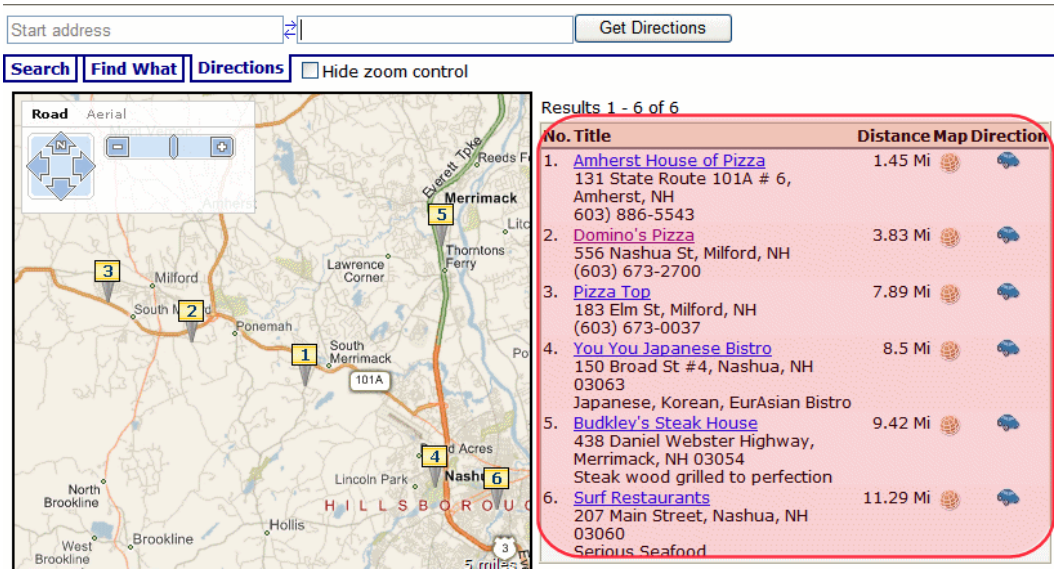
You can zoom the map in and out, get directions to a location, and narrow the list of locations using a text search. For example, if your map initially flags all schools in a geographic area, you can redraw the map to show only schools with a gym.

If you want the map to show events, you can apply dates to Ektron content, which allows searching by date or location.

IMPORTANT: As a map's boundaries change, only locations within boundaries appear. Likewise, if a date is assigned to content, only content within the selected date range appears.

If a map has at least one flagged location, a box can appear to its right with information about each location, which are sorted by distance to starting location. From this box, you can:

- view content to which the location is assigned
- find the distance to a location
- center the map on a location
- get directions to a location



- **No.**. Map item number. The lowest numbered item is closest to the map's center.
- **Title**. The content that corresponds to the numbered map location. Click the title to access that content on your website. The text below the title is the Metadata **Map Address** field, followed by the Metadata **Description** field (see illustration below).
- **Distance**. Distance from the map's center to this content location.
- **Map**. Click icon to recenter map on this content's location.
- **Directions**. Click icon to select the **Directions** tab and paste this content location's address into the end location box. You could then enter a starting location to get directions to this location.

When you hover the cursor over a numbered map location, a text "bubble" appears.

Bubble information:

- **Title**. Content that corresponds to the numbered map location. Click the title to proceed to that content on your website.
- **Summary**. Value of the content's **Summary** tab.
- **Metadata description**. Value of the content's **Metadata Description** field.
- **Get Directions**. Click **To Here** or **From Here** to select the **Directions** tab and paste this location's address into the To or From location box. Then, supply the missing location to get directions between locations.

Ektron supports 2 map providers: Google maps and Bing maps.

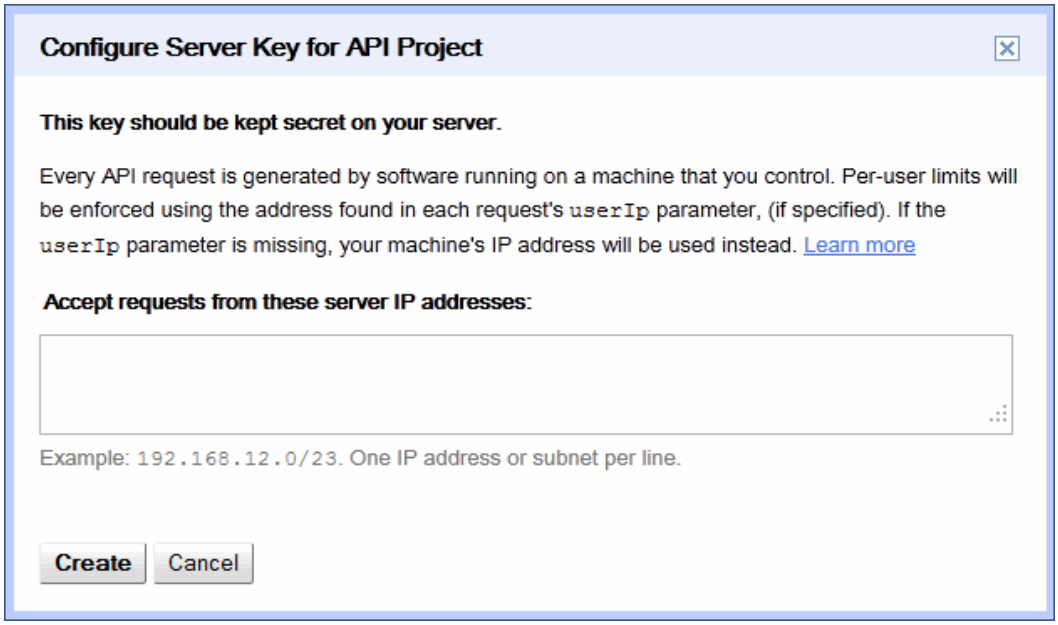
Configuring Google maps

Ektron provides a Google map key that you can use for testing on localhost. If you want to use Google map with your production server, you must install a license key.

Before your production server can use Google's map feature, follow these steps to obtain and install a license key. For Google Map's terms and conditions, see Google Maps/Google Earth APIs Terms of Service.

1. Go to Google APIs.
2. Log into (or create) a Google account.
3. Click **Services** from the left navigation panel.
4. Enable **Google Maps API v3**.
5. Click **API Access** from the left navigation panel.

6. Click **Create new Server key...**



7. Enter 1 or more server IP addresses and click **Create**. A key is created for server apps and browser apps (such as `AIzaSyBcN7RJzxr4f-iOOfdMwUq9op-5xrtOh4M`).
8. Without closing that page, open the *site root*`/web.config` file.
9. Go to the line that begins `<add key="GoogleMap"`.
10. Within that line, replace the following `localhost`key with the key you obtained in Step 7.

```
key="AQWESyCAtPrvLE6I2RTS7xK1-3btrrTO0eO6543"/>
```

Configuring Bing maps

9.10 and higher

Follow these steps to use Bing Maps with the Map server control.

1. Obtain a Bing Map API key (https://www.bingmapsportal.com/).
2. Insert the API key in `web.config`.

```
<add key="BingMapsAPIKey" value="nnnnnnnnnnnnnnnnnn"/>
```

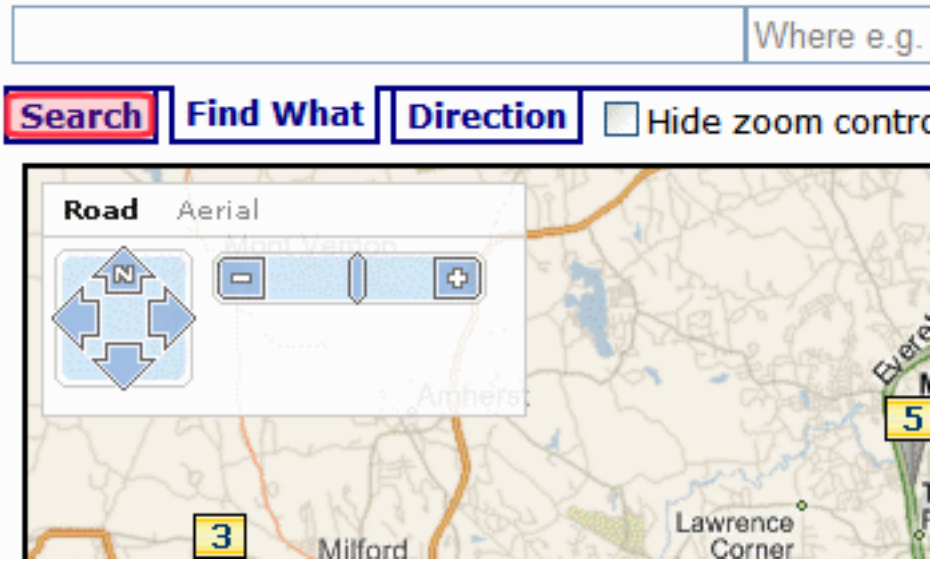
Controlling the map experience

NOTE: Within Visual Studio, you cannot see the map in design mode. But, you can right click the mouse and select **View in Browser** to see the effect of changing properties in a browser.

To learn about customizing the display using the Ektron Markup Language, see *map.ekml* on page 2637.

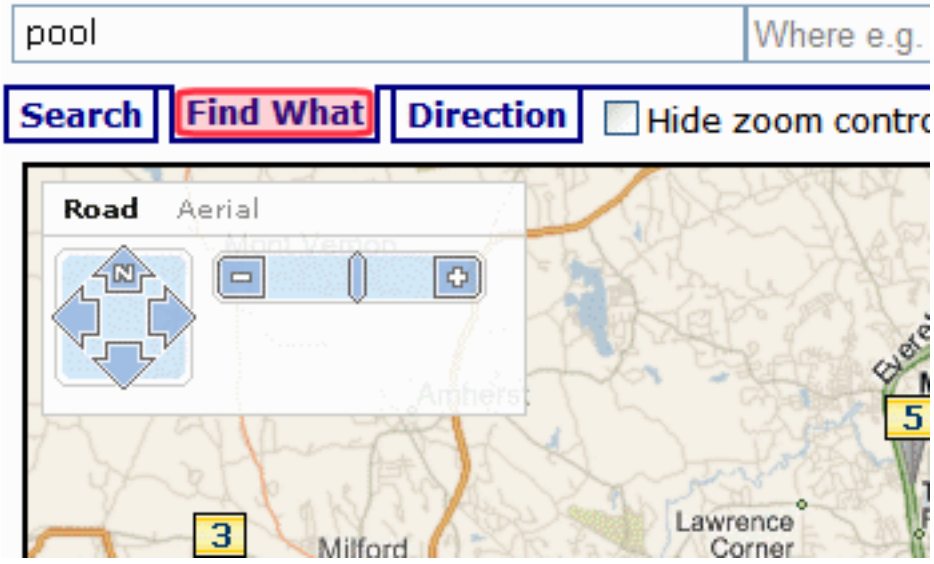
Recentering the map

The **Search** tab lets you recenter the map on a location. You can enter a combination of street address, city, state and zip code and click **Search**. Only content within that geographic area which satisfies other search criteria appears.



Finding Locations with a search term

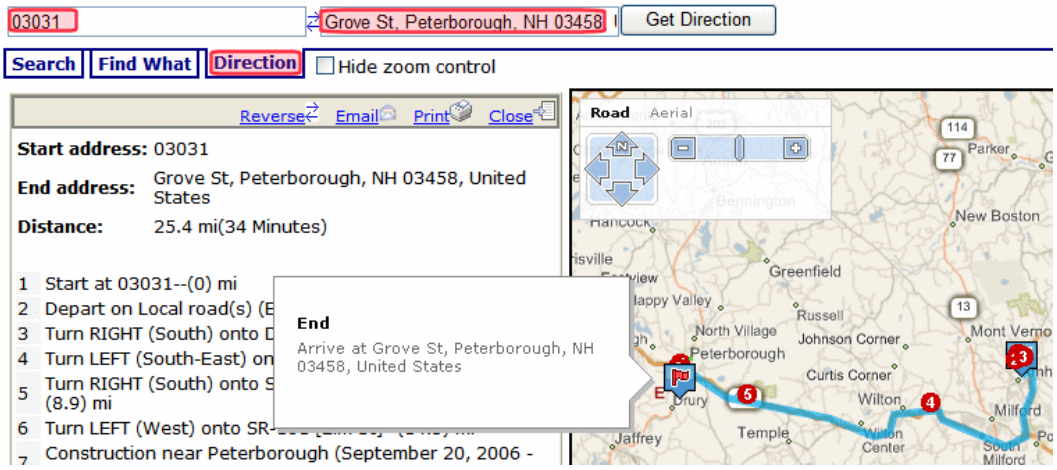
You can use the **Find What** tab to find only locations that include a search term. For example, if a map flags hotels, you can click the **Find What** tab then insert **pool** in the text box above to view only hotels with a pool. The **Find What** tab uses the same logic used in the Web Search to find content on your site.



Getting directions

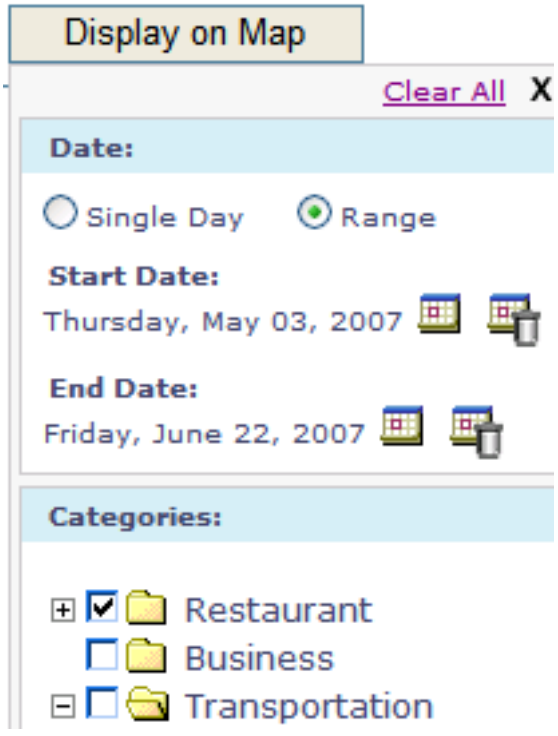
Use the **Directions** tab to get directions between 2 locations. Enter a combination of street address, city, state, and zip code into both text boxes above the tabs and click **Get Directions**. The screen displays text directions on the left, and a map of the directions on the right.

NOTE: To enable the emailing of directions for Bing maps in 9.10 and up, you must configure the system email and SMTP settings. See also: Enabling Email Notification



Restricting locations to taxonomy categories

If you use Ektron's taxonomy feature to classify mapped content, you can click **Display on Map** to restrict the map's locations to content in your categories. The popup window also prompts you to select one or a range of dates, and only retrieves content to which one of the selected dates is assigned.



For example, the category **Restaurant** has 5 subcategories in Ektron's sample site.

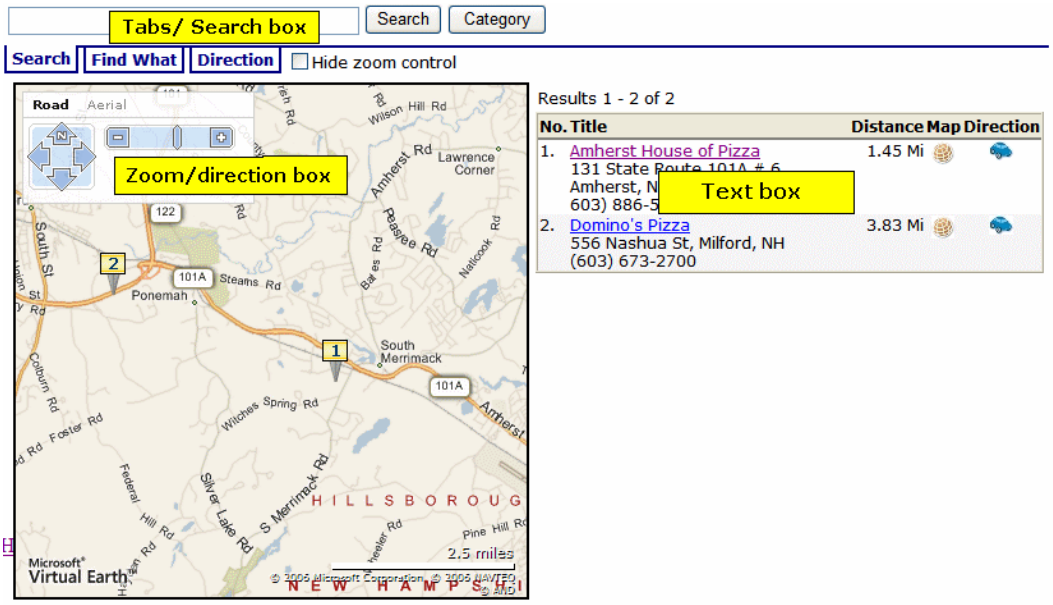
When you use the map, you can click **Display on Map**, see your content's categories, and select those of interest. The map updates to show content in selected categories only. To learn about assigning taxonomy categories to content, see Organizing Content with Taxonomies.

NOTE: While an OR logical relationship among selected categories is the most intuitive and common, you can set up AND or NOT relationships among categories.

Displaying/suppressing map elements

You can use Map properties to display or suppress the following map elements.

- Tabs/Search box. GeoControl
- Text box. DisplayTextResult
- Zoom/direction. ZoomControl; for Bing Maps in 9.10 and up, or for Enterprise maps, this property also controls the Map type selection.
- Map type (road, satellite, combination). TypeControl; affects only Google maps.



Setting a map's initial boundaries

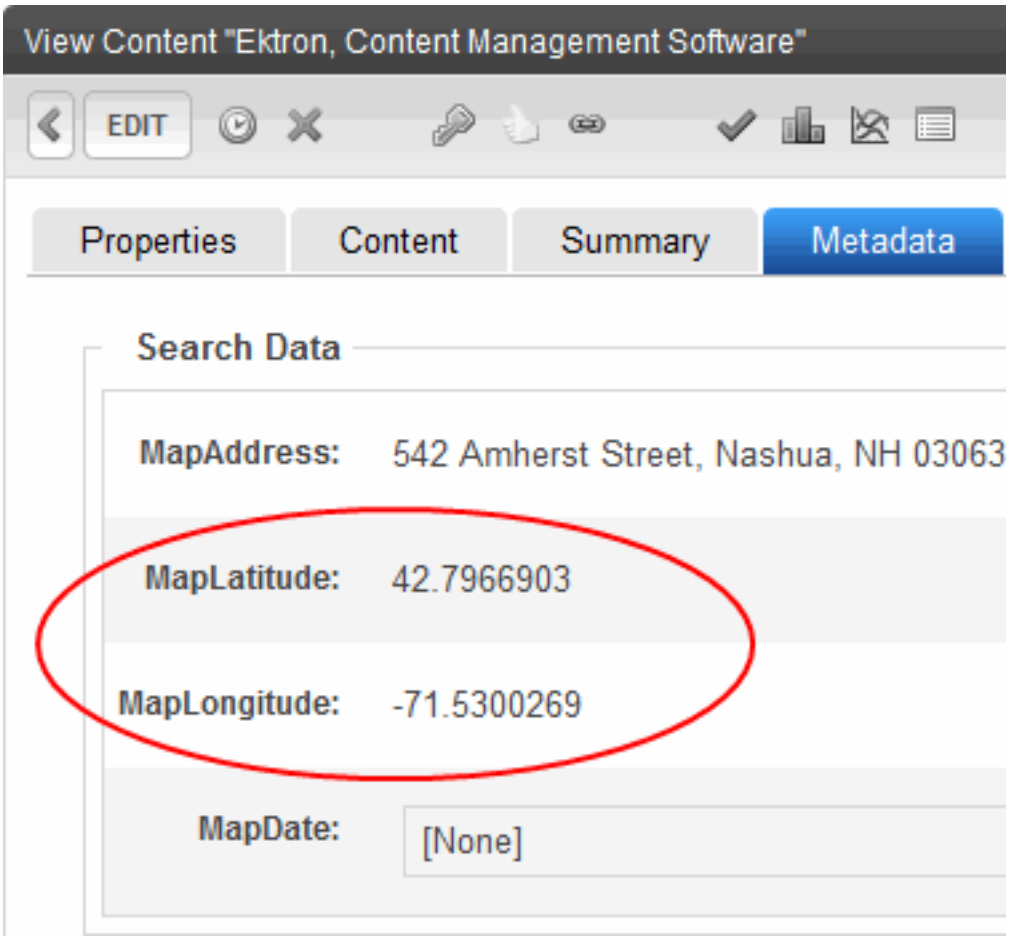
Because the map only displays content whose address lies within the map's boundaries, focus the initial display on your businesses/locations. Map properties let you specify a beginning address (or longitude/latitude) and a starting zoom level. All content with address data within that area is flagged on the map.

If your locations are too spread out to appear on a single map, create several regional maps. Each map server control must appear on a separate Web form.

IMPORTANT: You cannot place more than one map server control on a Web form.

Setting content found on a map

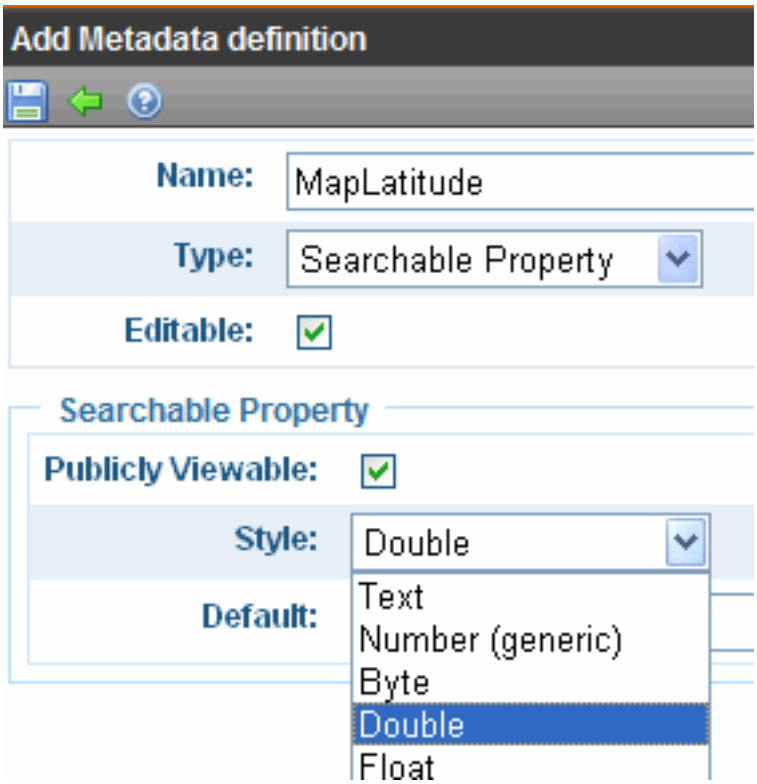
Whether you use Google or Bing Maps for Enterprise maps, content must have latitude and longitude values to appear on a map.



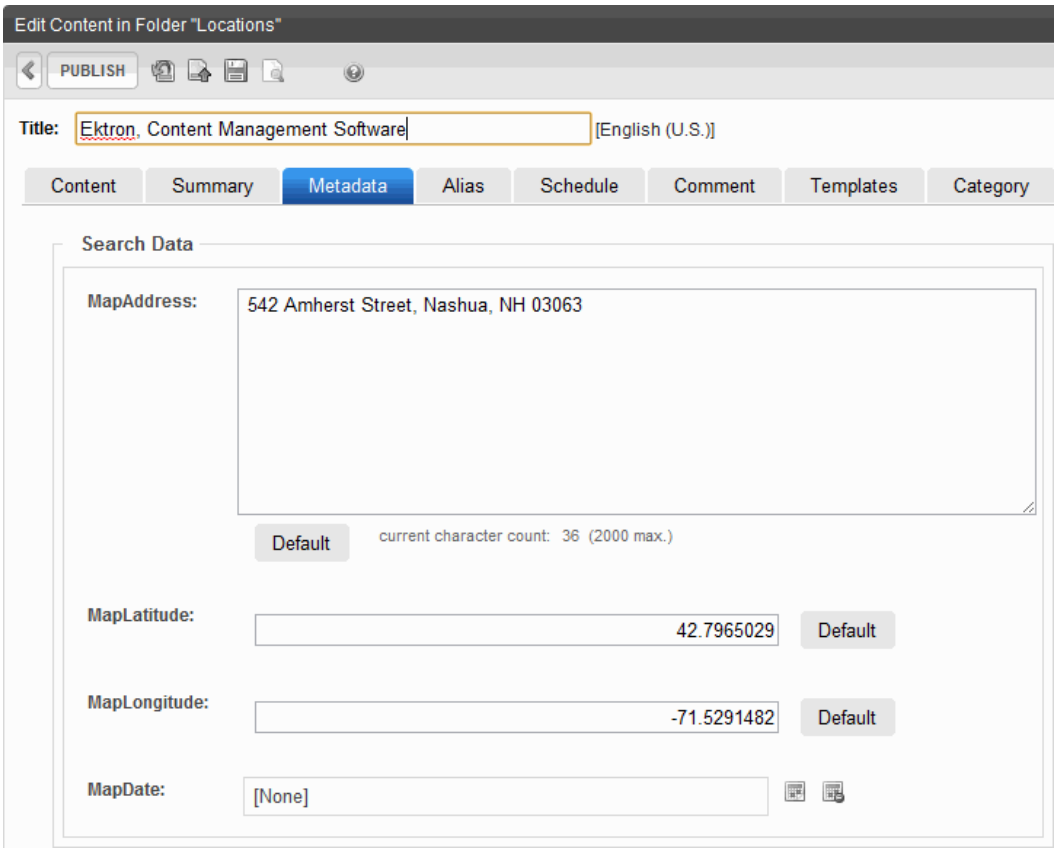
Google maps take a content item's address and return its latitude and longitude. You don't need to use Google's automatic retrieval of latitude and longitude. Instead, you can enter the values manually. To do so, open the content item, go to its metadata, and enter the latitude and longitude values under **Search Data**.

To automatically obtain latitude and longitude information for a content item:

IMPORTANT: The following procedure assumes you are using Ektron's sample site. If you are using the Min site, you must create searchable metadata definitions for **Map Address**, **Map Latitude** and **Map Longitude**. When defining **Map Latitude** and **Map Longitude**, set their **Style** to **Double**. If you are using dates with metadata, set **MapDate**'s style to **Date**.



1. Obtain a Google maps license key. See *Configuring Google maps* on page 2473
2. Create the content if necessary. On the content's **Metadata** tab, under **Search Data**, enter a **MapAddress** (that is, any combination of street, city, state, and zip code).



If you enter only a zip code, the latitude and longitude are set to that post office.

3. Publish the content.

   This action creates the Web services call to Google maps, which retrieves the latitude and longitude for each address. If Google maps cannot find a latitude and longitude (usually due to insufficient or conflicting information), it writes failure information to your server. You can view this under **Windows Event Viewer > EktronLog**. Any event's properties explain why the retrieval of latitude and longitude failed.

Restricting content for a particular map

You may want a map to show a subset of all content with a latitude and longitude. For example, your business includes restaurants and bakeries, and you want a map to show only bakeries.

To accomplish this, place content for restaurants in one folder, and bakeries in another. Then, in the map server control that shows bakeries only, at the `FolderID` property, identify the bakeries folder.

If you want another map to show both restaurants and bakeries, create the restaurant and bakery folders under a parent folder. Then, in that map server control's `FolderID` property, identify the parent folder, and set the `Recursive` property to **true**.

Membership

8.50 and higher

The Membership server control has a tabbed form that lets a site visitor create or update a profile. The control also can present new membership users with terms and conditions for using Discussion Boards when needed. See also: Membership Users and Groups.

NOTE: You can view the Membership server control on the OnTrek starter site when you edit your profile.

IMPORTANT: You cannot use the Membership server control with CMS users.

Users specify the following information in the Membership server control tabs.

Inserting the Membership server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

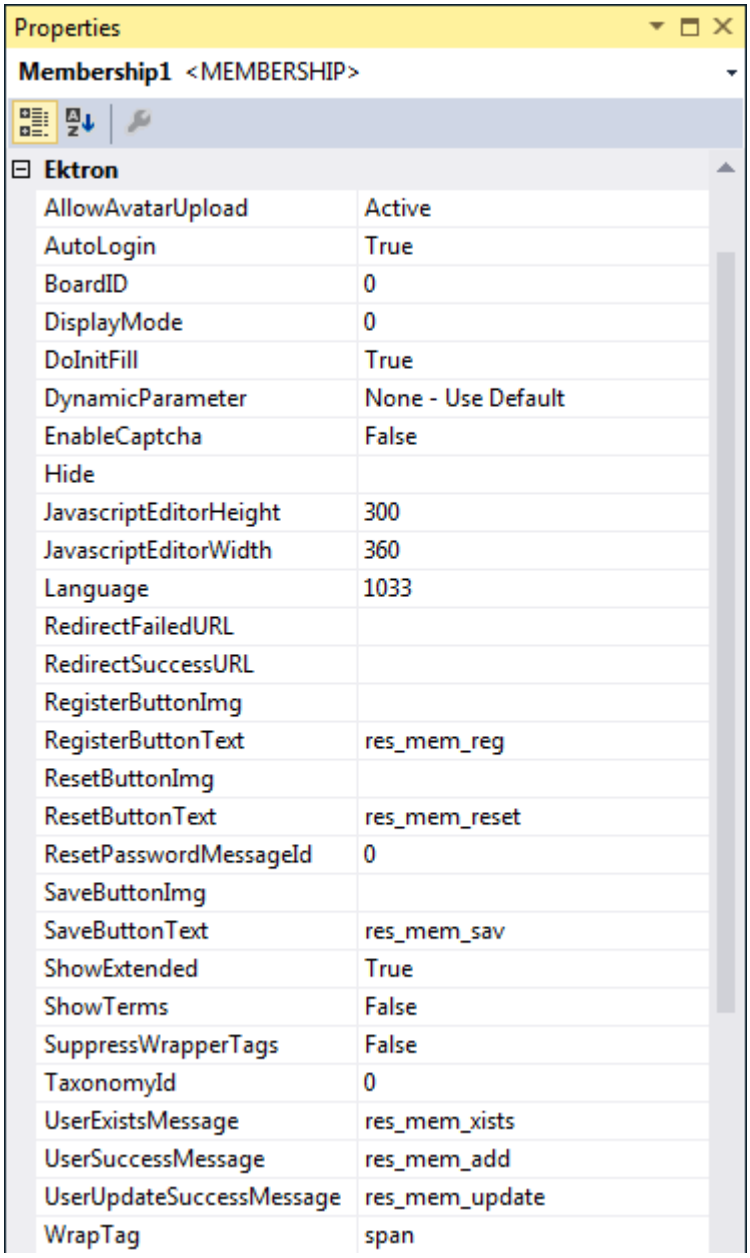
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Membership** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:Membership ID="Membership1" runat="server" />
   ```

4. Click on `Membership` in the code to display and modify the control's properties using the properties window of Visual Studio. The page is updated as you modify

the property values.



Membership properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AllowAvatarUpload** (String)

  When set to **Active**, **Click to upload your avatar** appears on the **General** tab. This allows site visitors to upload an avatar when registering for the site. When set to **Disable**, the link is hidden. To force site visitors to choose from a gallery of avatars, enter a path to the template containing them.

- **BoardID** (Long)

  The ID of the Discussion Board for which to show the Terms and Conditions. If you don't know the ID, click **Ellipses**, then sign in, browse to and select the Discussion Board.

- **DisplayMode** (Mode)

  Lets a developer decide what type of membership form is added to the Web form. The following describes possible settings.

  - **User Registration**. Lets a site visitor register as a membership user. Also allows membership users to update their information and preferences when logged in.
  - **Reset Password**. Lets a membership user reset a password.

    NOTE: Starting in Ektron 9.10, password restrictions defined in the Application Setup screen's **Regex** tab are not applied when memberships users reset their password. See also: Password Regex Tab.

    See also: Ektron Knowledge Base article "Changing the Membership User Password Reset Message".
  - **Unsubscribe Secured**. Lets membership user unsubscribe by entering a username and password.
  - **Unsubscribe Unsecured**. Lets membership user unsubscribe by entering a username.
  - **Account Activate**. Let a site visitor activate a membership by entering the ID number sent in the account verification email.

- **DoInitFill** (Boolean)

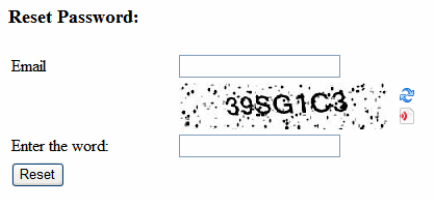
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter**

  To make this control dynamic, select id. When you do, this server control is attached to the user passed as a URL parameter.

- **EnableCaptcha** (Boolean)

  Use to add a Captcha control for more security. It looks like this.

  - **True**. Show Captcha
  - **False**. Hide Captcha

If this property is set to **true**, Captcha appears while the control is in **User Registration**, **Account Activate**, or **Reset Password** mode. See www.captcha.net for more information about Captcha.

- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **JavascriptEditorHeight** (Integer)
  Set the height in pixels for the eWebEdit400 editor. The default is **300**.
- **JavascriptEditorWidth** (Integer)
  Set the width in pixels for the eWebEdit400 editor. The default is **360**.
- **Language** (Integer)
  Set a language for the server control. This property shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **RedirectFailedURL** (String)
  The URL to which a membership user is sent if the registration fails.
  - If the page resides in the same folder as the registration page, enter the page name. For example, **RegFailed.aspx**.
  - If the page resides in a subfolder, enter its path. For example, **members\RegFailed.aspx**.
- **RedirectSucessURL** (String)
  The URL to which a membership user is sent when the registration succeeds.
  - If the page resides in the same folder as the registration page, enter the page name. For example, **RegSucceed.aspx**.
  - If the page resides in a subfolder, add its path. For example, **members\RegSucceed.aspx**.
- **RegisterButtonImg** (String)
  Lets you replace text on the register button with an image. Enter a path to the image. For example:

  ```
  http://www.example.com/buttons/registerbutton.gif
  ```

  If the image is located in the site root, enter only the subfolder path and image name. For example: `/buttons/registerbutton.gif`
- **RegisterButtonText** (String)
  Text that appears on the Register button. The default is **Register**. If you use a register button image, you do not see this text.
- **ResetButtonImg** (String)
  Lets you replace text on the reset button with an image. Enter a path to the image. For example:

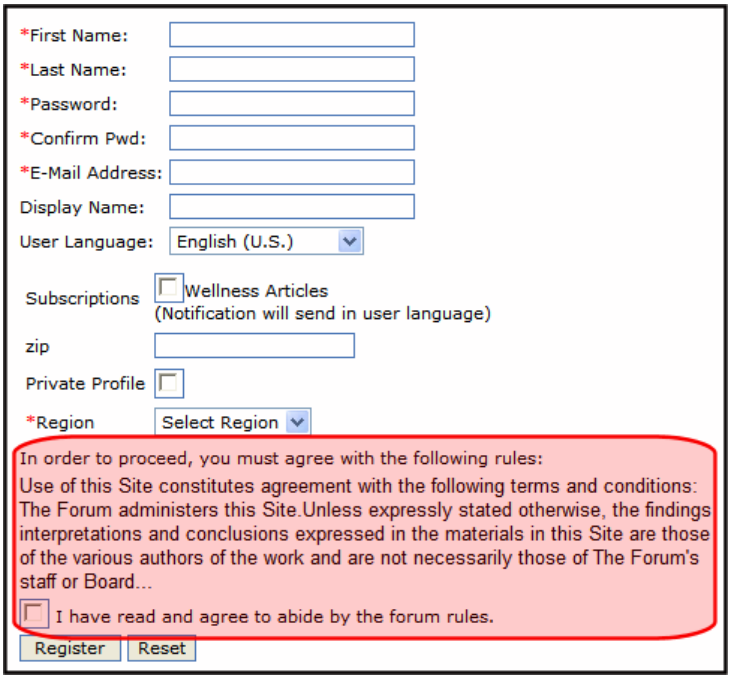
  ```
  http://www.example.com/buttons/resetbutton.gif
  ```

If the image is located in the site root, enter only the subfolder path and image name. For example: `/buttons/resetbutton.gif`

- **ResetButtonText** (String)
  Text that appears on the reset button. The default is **Reset**. If you use a reset button image, you do not see this text.
- **SaveButtonImg** (String)
  Lets you replace text on the Save button with an image. Enter a path to the image. For example:

  ```
  http://www.example.com/buttons/savebutton.gif
  ```

  If the image is located in the site root, enter only the subfolder path and image name. For example: `/buttons/savebutton.gif`
- **SaveButtonText** (String)
  Text that appears on the save button. The default is **Save**. If you use a save button image, you do not see this text.
- **ShowExtended** (String)
  Decide if the Custom User Properties Tab is available when using this control. The default setting is **True**.
  - **True**. Show Custom User Properties tab.
  - **False**. Hide Custom User Properties tab. For more information, see Creating Custom User properties.
- **ShowTerms** (Boolean)
  Terms and Conditions are defined for the Discussion Forum specified in the Membership server control's `BoardID` property. If you set the `ShowTerms` property to `True`, the Terms and Conditions for the forum identified in the control's `BoardID` property appear near the bottom of the Membership server control. A membership user must check the box to complete a registration.



If Terms and Conditions are defined for the specified Discussion Board and `ShowTerms` property is set to `False`, you must accept the Terms and Conditions the first time you create a post or a reply.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **TaxonomyID** (Long)

  The ID of a taxonomy available to users. A user editing or creating a profile can select which categories from this taxonomy to associate with their profiles.
- **UserExistsMessage** (String)

  The message that appears when a membership user already exists. The default message is: **Username(email) already exists!**
- **UserSuccessMessage** (String)

  The message that appears when a membership user successfully registers. The default message is: **You have registered successfully**. You can also use this property to set the message that appears when users successfully unsubscribe or reset their password. To do this, enter the success message in this property and set the `DisplayMode` property to the proper usage.
- **UserUpdateSuccessMessage** (String)

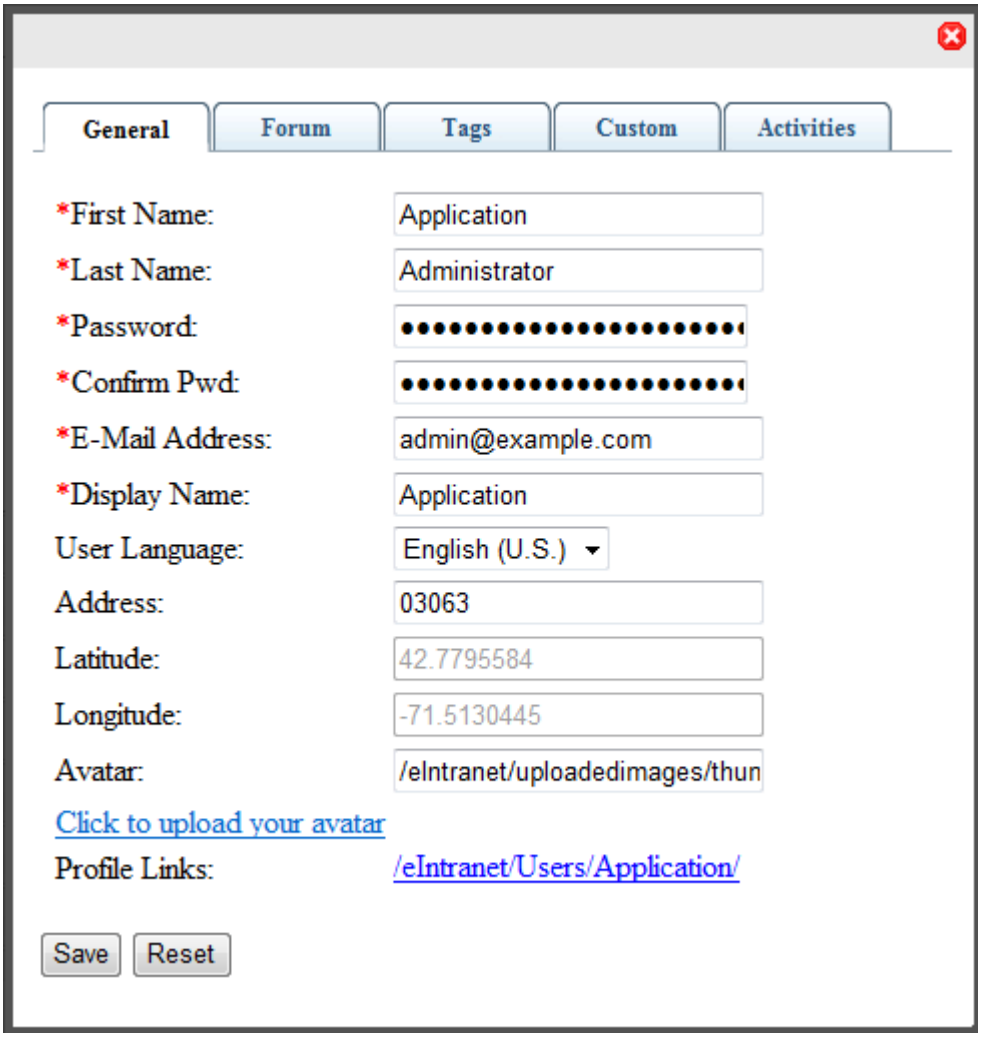
  The message that appears when a membership user successfully updates their information. The default message is: **You have successfully updated your information.**
- **WrapTag** (String)

  Lets a developer specify a server control's tag.

- **Span** (default). Designate an inline portion of an HTML document as a span element.
- **Div**. Apply attributes to a block of code.
- **Custom**. Lets you use a custom tag.

General tab

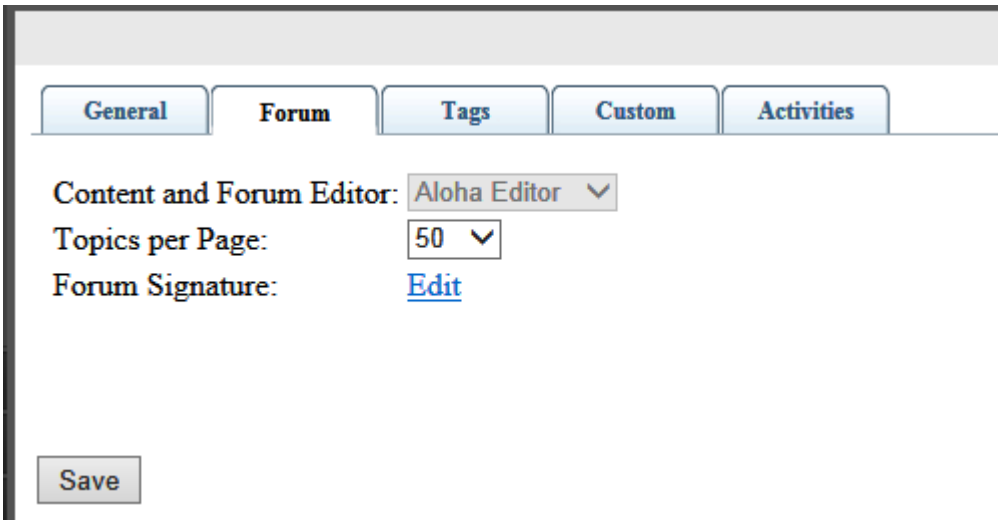
Fields with a red asterisk (*) are required fields.



- **First name**. Enter your first name.
- **Last Name**. Enter your last name.
- **Password**. Enter a password.
- **Confirm Password**. Re-enter the password.
- **E-Mail Address**. Enter your email address. Notification email is sent to this address unless the **Disable E-mail Notification** field is checked. Also, this address identifies the user sending Instant email.
- **Display Name**. Enter the name you want to display to others, which can be a nickname or title.
- **User Language**. Select from available languages.

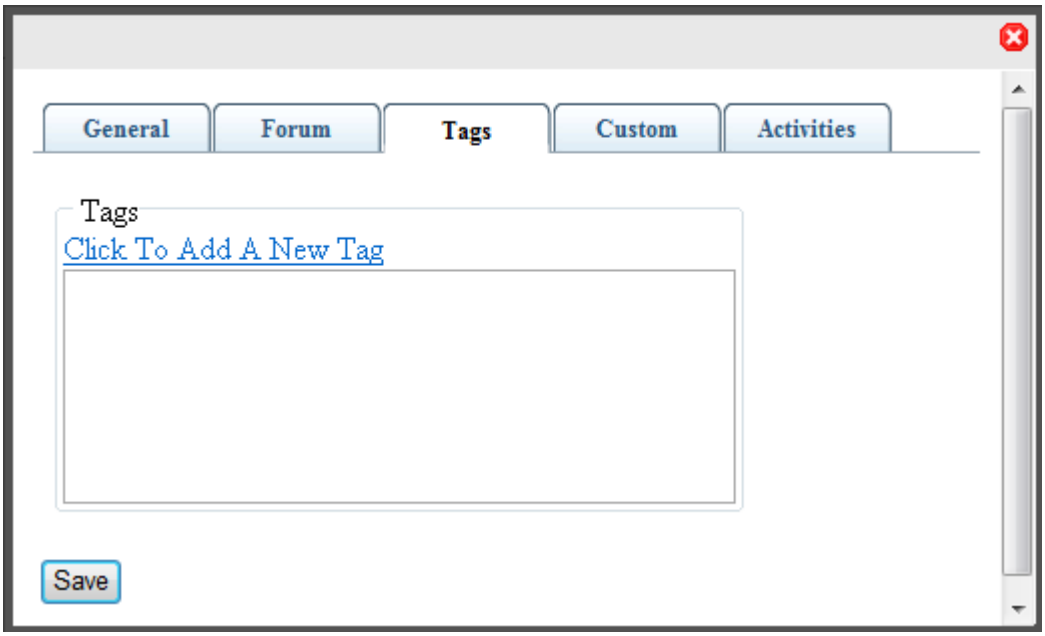
- **Address**. Enter the address (or just the zip code) of where you are located.
- **Latitude**. This field is automatically determined by the address.
- **Longitude**. This field is automatically determined by the address.
- **Avatar**. Click **Click to upload your Avatar** and choose an image file that you want to associate with your profile. By default, the maximum file size of an avatar is 200 kilobytes. Also, the avatar's height and width cannot exceed 125 pixels preserving the aspect ratio.

Forum tab



- **Content and Forum Editor**. If the Ektron administrator has chosen an editor, it appears but cannot be changed. Otherwise, possible editors appear in a drop-down, and you can choose one. See also: Changing the Default Editor.
- **Topics per Page**. Select a number of topics to display on a page. If the number of topics exceeds the number you select, a scroll bar lets you see the additional topics. For example, if you select 10, and there are 25 topics, only 10 are displayed at one time, but you can scroll down to see all topics.
- **Forum Signature**. Click **Edit** to enter or modify a signature that appears below each post you make to a forum topic.

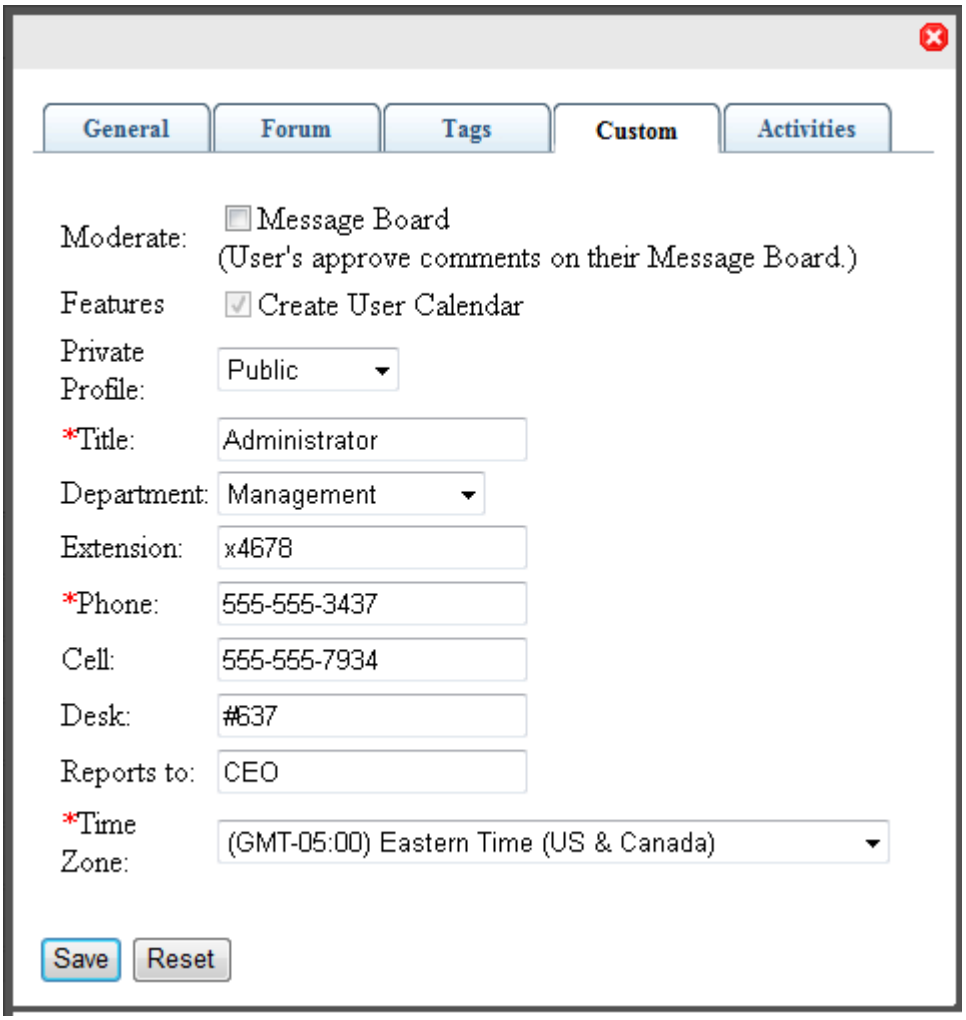
Tags tab



- **Tags**. Keywords that you can assign to content and library items, which allows for tag-based searching. For example, you can add the tag **EAC** (Employee Activity Committee), and tag users related to the EAC. In this way, you can search for the users to which the **EAC** tag is applied. A tag cannot exceed 25 characters, and can include only alphabetical or numeric characters, a hyphen, or an underscore.

Custom tab

Fields with a red asterisk (*) are required fields.



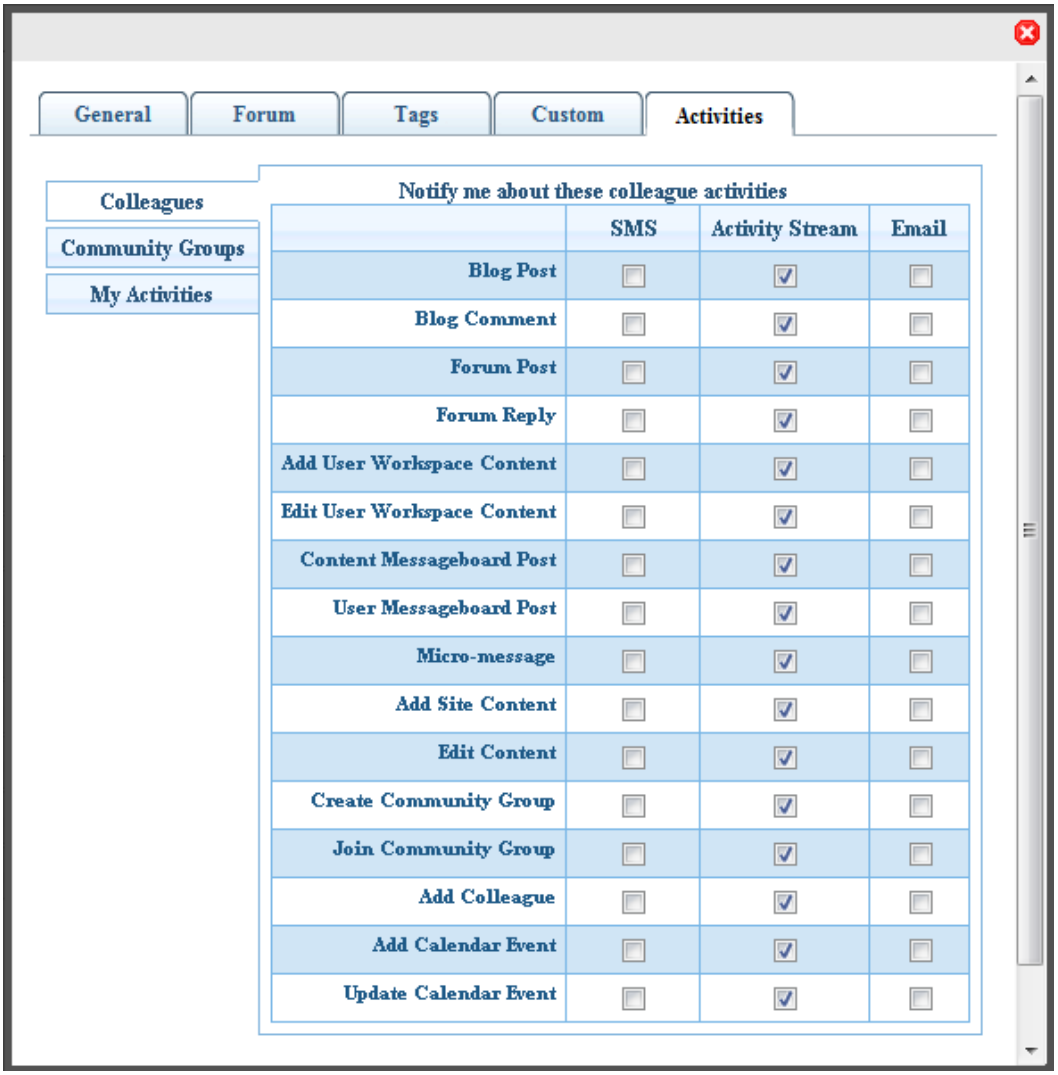
- **Moderate**. Check this box to give this user privileges on a message board to approve and delete posts. Regular users in a group message board can create and delete only their own posts.
- **Features**. Check this box to give the user an individual calendar. (There may be other features to grant also.)
- **Private Profile**. Choose an option:
  - Public. User information is accessible by others.
  - Private. User information is not accessible by others.
  - Colleagues. User information is accessible only by colleagues of the user.
- **Title**. Enter the title of the user.
- **Department**. Select the department to which this user belongs. The list contains any departments that you have created.
- **Extension**. Enter the user's telephone extension.
- **Phone**. Enter the user's company telephone number.
- **Cell**. Enter the user's cell phone number.

- **Desk**. Enter the value that identifies the location of the desk (or cubicle, or office), if your office identifies such things. This can be valuable in locating an employee on an office map.
- **Reports to**. Enter the person to whom the user reports. This can be valuable for developing organizational charts.
- **Time Zone**. Select the time zone where the user works.

Activities tab

By default, all activity is checked for the Activity Stream widget.

Check boxes to select criteria for activity types you want to see from colleagues or community groups. You can also select activity that you want colleagues to see, under the **My Activities** subtab. See also: Sending Notifications to a Community.



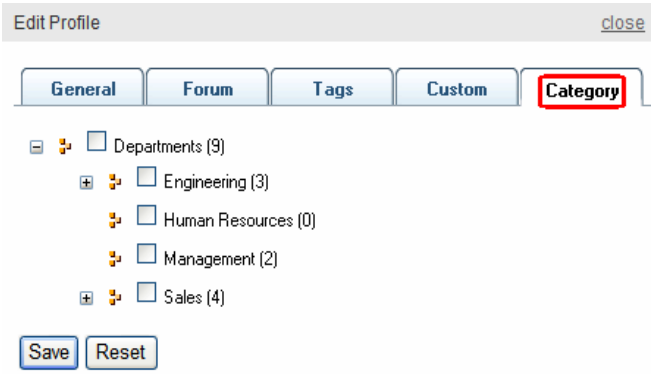
- **SMS**. A checked box indicates that the activity type will be sent to your cell phone Short Message Service (SMS).
- **Activity Stream**. A checked box indicates the activity type that will appear in the Activity Stream widget.

- **Email**. A checked box indicates the activity type that will be sent to your Email address, which is specified in your profile's General tab.

Category tab

The Category tab lets you select from a list of taxonomy categories with which you want to be associated. You determine which taxonomy appears on the tab by setting the `TaxonomyID` property.

IMPORTANT: The Category tab appears only when an ID is assigned to the `TaxonomyID` property.

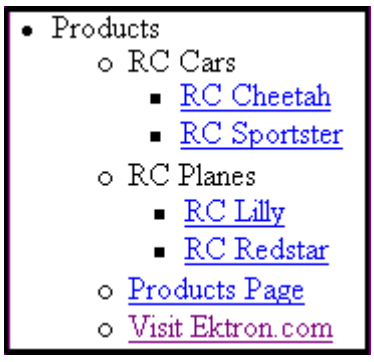


Menu

8.50 and higher

IMPORTANT: Starting with Release 8.6, the Menu server control was replaced by the FrameworkUI: <ektron:MenuView> templated server control. If you are already using the Menu server control, you can continue to do so, but Ektron recommends using current versions of functionality.

The Menu server control calls a menu to be displayed on a page. Using the Menu server control, you can manipulate a menu by using the `DisplayXslt` property. Below is a menu display with the SampleMenu XSLT.



Before you can use the Menu server control, you must create a menu in the Workarea.

This section also contains the following topics.

Inserting the Menu server control onto a page

PREREQUISITE

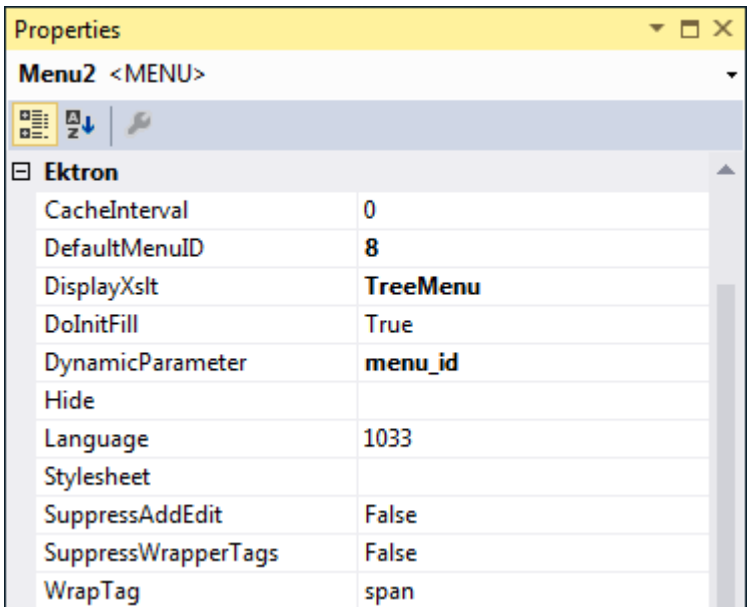
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Menu** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Menu ID="Menu1" runat="server" />
```

4. Click on `Menu` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



Menu properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual

Studio® help.

- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **DefaultMenuID** (Long)
  The ID of a menu that appears where you insert this server control if no other menu is identified or available. If you don't know the ID number of the menu, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147
- **DisplayXslt** (String)
  The XSLT to use to render the menu.
  - **None**. Databind only
  - **SampleMenu**. A sample display, formatted as a bulleted menu list
  - **TreeMenu**. A sample display, formatted as a folder tree. You can expand the tree by clicking on the folder icon.
  - **Path to Custom Xslt**. If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  For more information on using the SampleMenu and TreeMenu DisplayXslt, see *Using DisplayXslt samples* on the next page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)
  Select **menu_id**. When you do, this server control uses the menu passed as a URL parameter.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

For more information, see Creating a Menu in Another Language.

- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control.

  Leave blank to use the default style sheet.

  To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.

  The default style sheet is `\webroot\siteroot\Workarea\csslib`.

- **SuppressAddEdit** (Boolean)
  - **True**. Suppress the Add and Edit buttons when a user is logged in.
  - **False**. Show the Add and Edit buttons when a user is logged in.

  The default is False.

- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

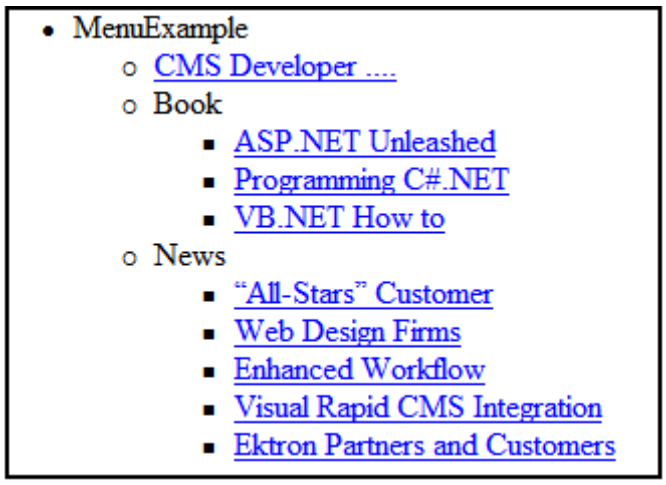
Using DisplayXslt samples

Two DisplayXslt samples are provided with the Menu server control, SampleMenu and TreeMenu. This section explains how to use them.

WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

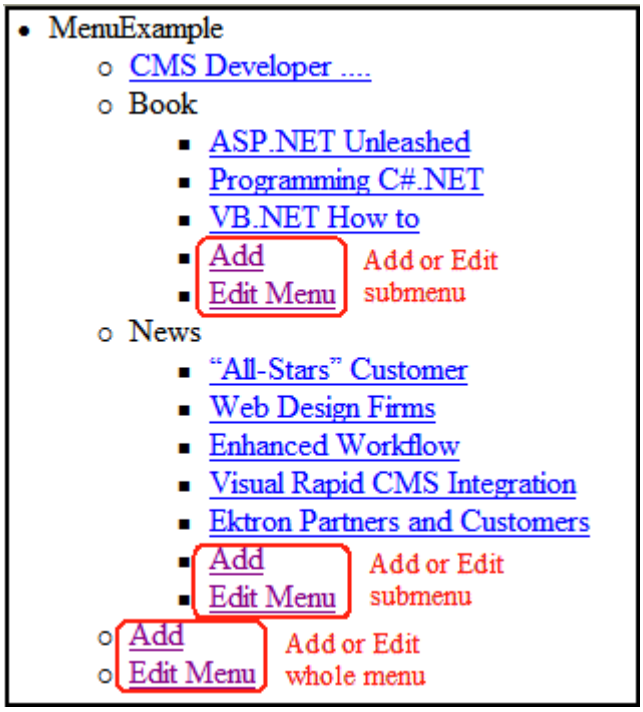
SampleMenu

The SampleMenu DisplayXslt lets you display an Ektron menu as a bulleted item list. See below for an example.



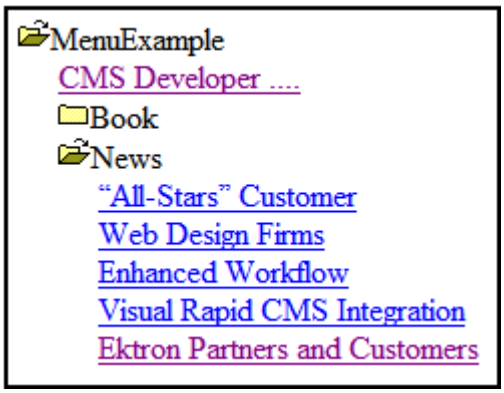
When you are logged in to your Ektron site, 2 menu items are added to each section of the menu: Add and Edit Menu. The user can use these to add a new menu item or edit an existing one. See example below.

For information on a adding or editing a menu item, see Adding a Menu Item.



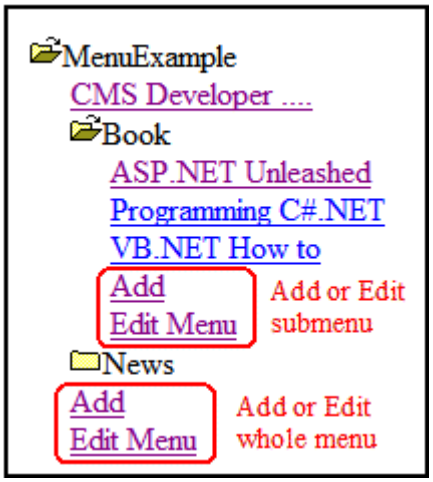
TreeMenu

The TreeMenu DisplayXslt lets you display an Ektron menu as a clickable folder list. You can expand and collapse the menu by clicking on the folder icons.



When you are logged in to your Ektron site, 2 menu items are added to each section of the menu: Add and Edit Menu. The user can use these to add a new menu item, or edit an existing menu item. See example below.

For information on a adding or editing a menu item, see Adding a Menu Item.



NOTE: When using the TreeMenu server control with Visual Studio and Windows 2003, the folder images do not display during design-time. They do, however, display correctly during run-time.

Retrieving the XML structure of a menu

Retrieving a menu's XML structure provides greater control over developing XSLs. The following is an example of how to retrieve the XML structure:

1. Open a new Web form.
2. Drag and drop a Menu server control onto it.
3. Set the `DefaultMenuID` properties.
4. Drag and drop a textbox on the Web form.
5. Set the `TextMode` property to MultiLine.
6. Set the width of the text box to at least 400px.
7. On the code-behind page, add the following line.

```
Textbox1.Text = Menu1.XmlDoc.InnerXml
```

8. Build the project.
9. View the Web form in a browser.
10. The XML structure of the menu appears in the textbox.

Metadata

8.50 and higher

The Metadata server control lets you add the metadata from content blocks to a Web page. This lets developers add metadata quickly without having to type it in. You can add metadata from a single content block, multiple content blocks, or dynamically pass a content ID from a URL.

In contrast, the MetaDataList server control lets you create a list of content blocks to display on your site, based on metadata assigned to the content. See also: *MetadataList* on page 2503.

Inserting the MetaData server control onto a page

PREREQUISITE

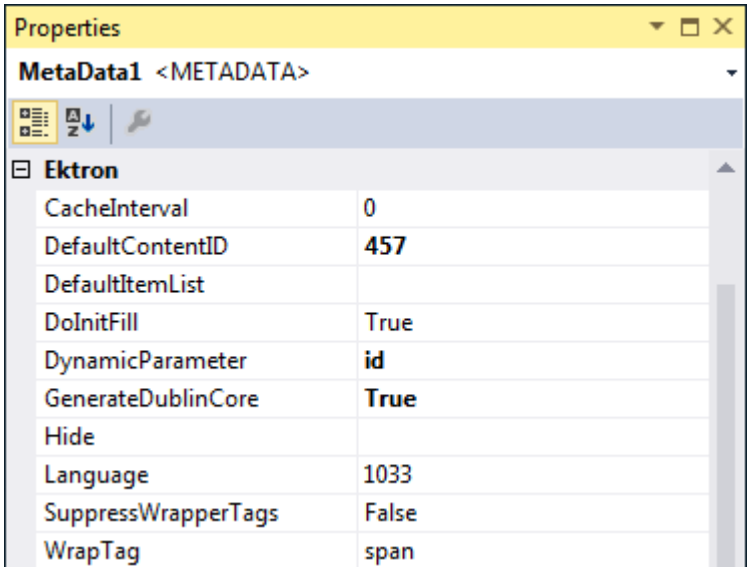
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **MetaData** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:MetaData ID="MetaData1" runat="server" />
   ```

4. Click on `MetaData` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



Metadata properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DefaultContentID** (Long)

  Enter the ID of the content block whose metadata is added to the page. If you don't know the ID number of the content block, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147. If you want to add metadata from several content blocks, set this property to 0 (zero) and use the `DefaultItemList` property to identify them.

- **DefaultItemList** (String)

  A bracket-separated list of content block IDs whose metadata added to the page. This list is used only if the `DefaultContentID` property is set to 0 (zero). For example:

  DefaultItemList **[92][12][4][7]**

  In the ID list, you can specify metadata definitions to *exclude* for each content block. To exclude a metadata definition, insert a semicolon after the ID and enter the metadata definition. For example,

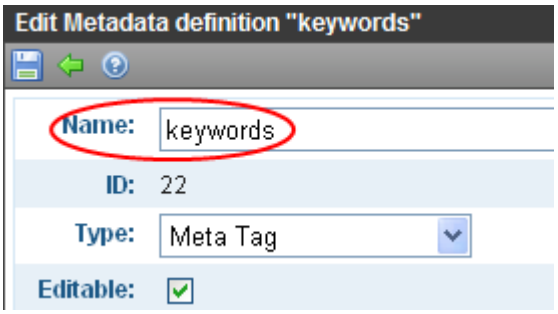
  DefaultItemList **[30][10;Title][23;Title;Description]**

  In the above example, the control will:

  - add all metadata definitions for content block 30
  - exclude the `Title` metadata definition for content block 10
  - exclude the `Title` and `Description` metadata definitions for content block 23

  Note the following criteria for metadata definitions that may be excluded:

  - the definition is case-sensitive, so must exactly match how the **Name** field of the Metadata Definitions screen.

  - The metadata definition type must be **Meta Tag**

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **DynamicParameter** (String)

  Gets or sets the QueryString parameter to read a content ID dynamically.

- **GenerateDublinCore** (Boolean)
  When enabled, this property automatically creates 7 of the Simple Dublin Core metadata fields from standard Ektron system properties. The default is false.
  - **True**. Generate Simple Dublin Core metadata fields
  - **False**. Do not generate Simple Dublin Core metadata fields

  The 7 fields and how they are associated with the Ektron properties is explained in Applying Simple Dublin Core Metadata
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

These steps show how to use the MetaData server control.

1. Drag a MetaData server control onto a template.
2. Set the properties of the Metadata server control. This will create the following HTML in the HTML body.

   ```
   <cms:MetaData id="MetaData1" runat="server"
   DefaultContentID="12"></cms:MetaData>
   ```

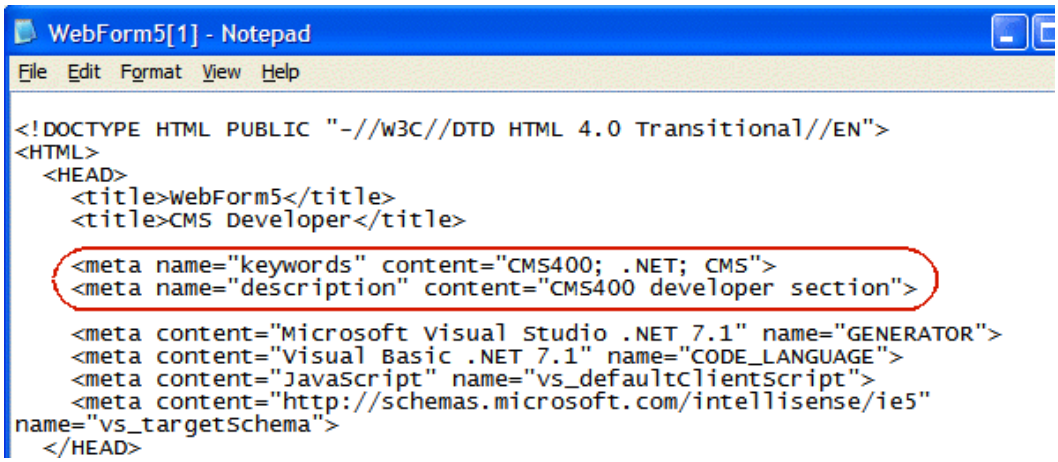
   Or, if you are using multiple content block IDs in the `DefaultItemList`, the following HTML is created.

   ```
   <cms:metadata id="MetaData1" runat="server" DefaultItemList="[12,7,4]">
     </cms:metadata>
   ```

3. Click the **HTML** tab and copy that line from the <body> tag to the <head> tag.

```
<HEAD>
    <title>WebForm5</title>
    <cms:MetaData id="Metadata2" runat="server" DefaultContentID="12"></cms:MetaData>
    <meta name="GENERATOR" content="Microsoft Visual Studio .NET 7.1">
    <meta name="CODE_LANGUAGE" content="Visual Basic .NET 7.1">
    <meta name="vs_defaultClientScript" content="JavaScript">
    <meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/i
</HEAD>
<body>
    <form id="Form1" method="post" runat="server">
    </form>
</body>
```

4. Save the Web form and rebuild the solution.
5. Open the Web page in the browser.
6. Right click the Web page and click **View Source**.
7. In the <head> tag, you see meta tags from the content block added to the page, as shown in the following image.



MetadataList

8.50 and higher

IMPORTANT: Starting from release 8.6, the MetadataList server control was replaced by the FrameworkUI: <ektron:ContentView> templated server control. If you are already using the MetadataList server control, you can continue to do so, but Ektron recommends using current versions of functionality.

The MetadataList server control creates lists based on Keyword Names and Keyword Values contained within content metadata. The list can display the information as a list of hyperlinks. You can choose, based on properties you set, to display the summary and how to order the display. See also: Working with Metadata.

NOTE: On a PageBuilder page, you can insert a metadata list using the MetaDataList widget. See also: Creating and Using Widgets.

Inserting the MetadataList server control onto a page

PREREQUISITE

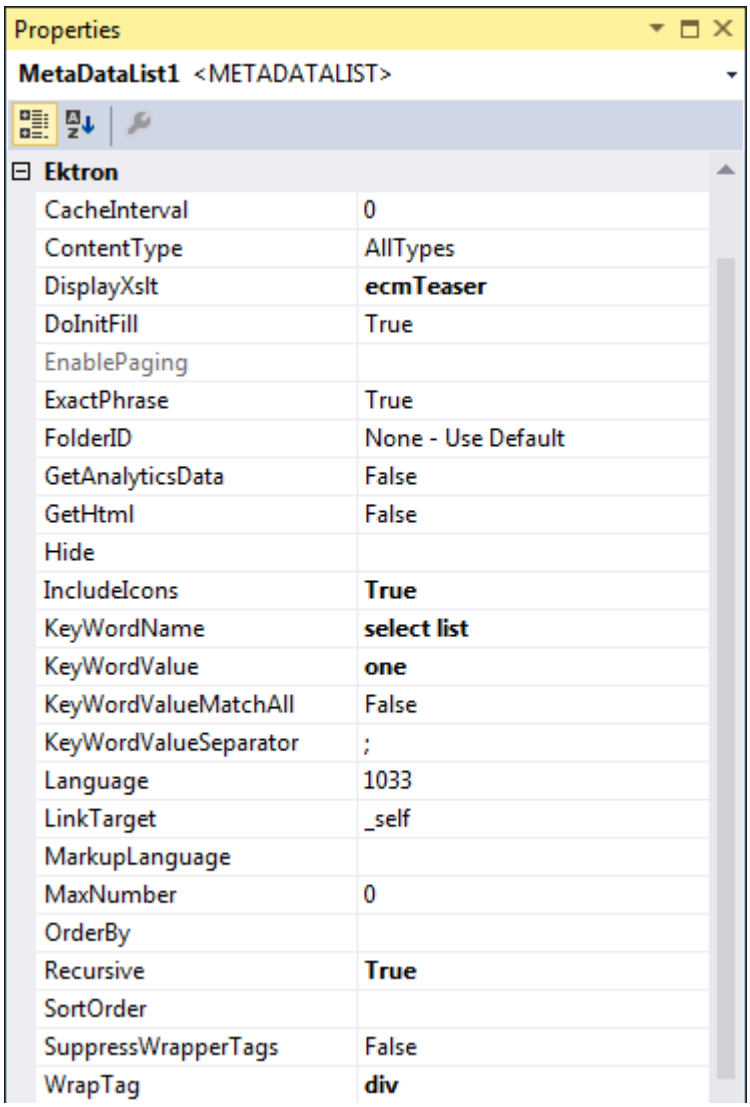
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **MetadataList** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:MetadataList ID="MetadataList1" runat="server" />
```

4. Click on `MetadataList` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



MetadataList properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

  IMPORTANT: If the `EnablePaging` property is set to True, the `CacheInterval` property is disabled.

- **ContentType** (Ektron.Cms.Controls.CmsWebService.CMSContentType)
  Select a type of content for this control. Choices are:
  - All Types
  - Content
  - Forms
  - Archive_Content
  - Archive_Forms
  - Assets
  - Archive_Assets
  - LibraryItem
  - Multimedia
  - Archive_Media
  - NonLibraryContent
  - DiscussionTopic
- **DisplayXslt** (String)
  Determines how the information on the page appears
  - **None**. Databind only
  - **ecmNavigation**. Lists the title of every content block in the folder.
  - **ecmTeaser**. Lists the title of every content block in the folder plus the content summary.
  - **Path to Custom Xslt**. If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

  WARNING!
  If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **EnablePaging** (Boolean)

  This property, in conjunction with the **MaxNumber** property, lets site visitors view an unlimited number of items while controlling the amount of screen space. The **MaxNumber** property limits the number of items displayed. If you set this property to **True**, and the number of items exceeds **MaxNumber**, navigation aids appear below the last item, allowing the visitor to go to the next screen. See example below.

  So, for example, if specified metadata is found in 9 items and the `MaxResults` property is set to 3, the screen displays only the first 3 items. When the site visitor clicks **[Next]**, the visitor sees items 4, 5 and 6.

  NOTE: If the `EnablePaging` property is set to True, the `CacheInterval` property is disabled.

- **ExactPhrase** (Boolean)

  Determines whether the KeyWordValue needs to match the metadata value exactly. For example, if "site" is the KeyWordValue, the title of a content block is "Welcome to the site" and ExactPhrase is set to **True**, you would not see the content block in the metadata list. This is because "site" does not equal "Welcome to the site".

  - **True**. Match exact phrase
  - **False**. Doesn't need to match exact phrase

- **FolderID**

  The folder ID from which content is retrieved. At the `Recursive` property, you determine if content in this folder's subfolders is also retrieved.

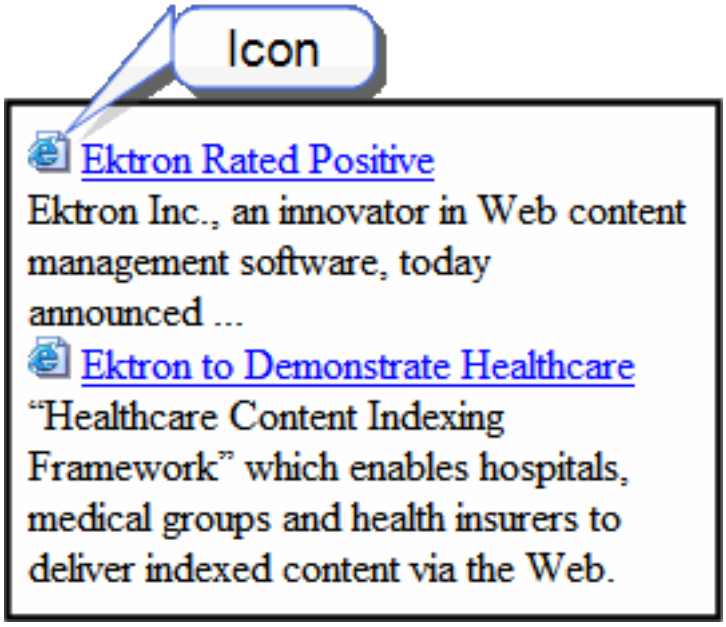
- **GetAnalyticsData** (Boolean)

  Set this property to **True** if you want the following information for each content in the list. Returns **Content View Count**, **Content Rating**, **Content Rating Average**. Create your own XSLT styles to display this data.

  IMPORTANT: This property provides reliable data only when the Business Analytics Feature is on. See Running Ektron Business Analytics.

- **GetHtml** (Boolean)
  Set to **True** if you want to display the content (html body) for all content to appear on this metadata list. For example, you want to display content inside a Web server control such as a GridView.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **IncludeIcons** (Boolean)
  Choose whether to display icons next to the metadata list's links.

  NOTE: This property only works when ecmSummary or ecmTeaser are used in the `DisplayXslt` property. When the `[$ImageIcon]` variable is used in an EkML file and that file is assigned to the `MarkupLanguage` property, this property acts as True.



- **KeyWordName** (String)
  KeyWordName represents a metadata definition, that is, the container for the KeyWordValues. Examples of a KeyWordName are **Keywords** and **Title**. If you are authenticated, you can click the ellipsis button and select from a list of existing metadata definitions. For information on creating metadata definitions, see Adding a Metadata Definition.
- **KeyWordValue** (String)
  Enter the values associated with the KeyWordName. Only content whose metadata (defined at the `KeyWordName` property) matches this value appears on the metadata list. Examples of a KeyWordValue are "home; page; company."

NOTE: The character that separates multiple items is defined at the `KeyWordValueSeparator` property. At the `KeyWordValueMatchAll` property, you determine if all metadata definition values must match or any one of them.

- **KeyWordValue MatchAll** (Boolean)

  This property is only used if you enter more than one keyword value. If you do, and only want content to appear on the metadata list if *all* values entered at the KeyWordValue field match their metadata values, enter **True**. If metadata can appear on the list as long as *any value* defined at the KeyWordValue field matches the selected metadata value for a content item, enter **False**.

  Example:

  KeyWordValue for **Title** (assigned for this server control): home; page; company. Metadata values for a content item's **Title** metadata definition field: software; ektron; company.

  - If `KeyWordValueMatchAll =true`, content does not appear on metadata list because some items do not match.
  - If `KeyWordValueMatchAll = false`, content item appears on metadata list because one item (company) matches.

- **KeyWordValueSeparator** (String)

  Enter the character used to separate the list of keyword values. An example is a semicolon(;).

- **Language** (Integer)

  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LinkTarget** (ItemLinkTargets)

  Determines the type of window that appears when you click a link in the server control.

  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.

- **MarkupLanguage** (String)

  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.See also: *Ektron Markup Language* on page 2633

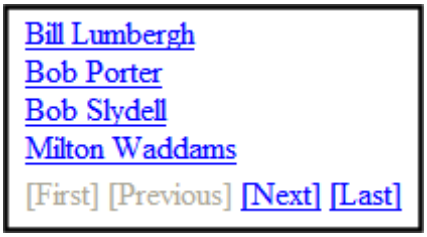
  See also: *metadatalist.ekml* on page 2640

NOTE: If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored. If the EkML file contains the `[$ImageIcon]` variable, the `IncludeIcons` property acts as True.

- **MaxNumber** (Integer)

  Enter the maximum number of items to appear in the initial display of this server control. To set no maximum, enter zero (0). To let site visitors view more than the maximum but limit the amount of space being occupied, enter the maximum number of results per page here. Then, set the `EnablePaging` property to **true**.

  If you do and more than the number of `MaxResults` are available, navigation aids appear below the last item to help the site visitor view additional items. See example below.

- **OrderBy** (Ektron.Cms.Controls.CmsWebService.ContentOrderBy)

  The order of the list to be returned.

  - **Title**. The title of the content block
  - **ID**. The content block ID
  - **Date Created**. The date the content block was created
  - **Date Modified**. The date the content block was last modified
  - **LastEditorLname**. The last editor's last name
  - **LastEditorFname**. The last editor's first name
  - **ContentRatingAverage**. Business Analytics Content Rating
  - **ContentViewCount**. Business Analytics Content Views

- **Recursive** (Boolean)

  Whether to search sub-folders of the identified root folder. The starting folder is identified in the `FolderID` property.

- **SortOrder** (String)

  Choose the order direction of the list, Ascending or Descending.

- **SuppressWrapperTags** (Boolean)

  This property is set to `false` because Ajax uses `<div>` tags to rewrite the region around the tag. You *cannot* change the value to `true`.

- **WrapTag** (String)

  Lets a developer specify a server control's tag.

  - **Span** (default). Designate an inline portion of an HTML document as a span element.

- **Div**. Apply attributes to a block of code.
- **Custom**. Lets you use a custom tag.

Retrieving the XML structure of a MetadataList

Retrieving the XML structure of XML content allows for greater control over developing XSLs. The following is an example of how to retrieve the XML structure.

1. Open a new Web form.
2. Drag and drop a MetadataList server control onto it.
3. Set the `KeyWordName` and `KeyWordValue` properties.
4. Drag and drop a textbox on the Web form.
5. Set the `TextMode` property to **MultiLine**.

   NOTE: Set the text box width to at least 400px.

6. On the code-behind page, add the following line.

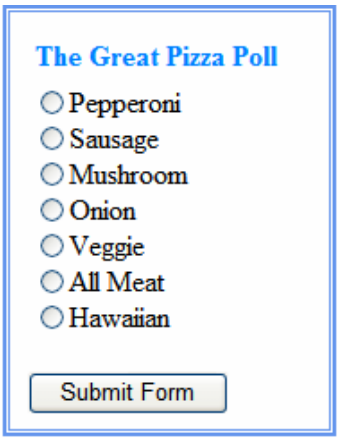
   ```
   Textbox1.Text = Metadata1.XmlDoc.InnerXml
   ```

7. Build the project.
8. View the Web form in a browser.
9. The XML structure of the MetadataList appears in the textbox.

Poll

8.50 and higher

The Poll server control displays a poll or survey created from an Ektron form on a Web page. When added to a template and visited, the poll might look like this. You can change a poll's appearance by modifying its properties.



You should display a poll or survey with a Poll server control, because it provides great flexibility with the poll's appearance. Typically, developers want a poll or survey in a small section of a Web page, not the main content. By using the

`EnableAjax` property, you can display the results in the same area as the poll or survey without refreshing the entire page.

However, if you want the form/poll/survey's response to be either **Redirect to a file or page** or **Redirect form data to an action page**, you must use a Form Block server control to display the form on a Web page. See also: *FormBlock* on page 2433.

Inserting the Poll server control onto a page

PREREQUISITE

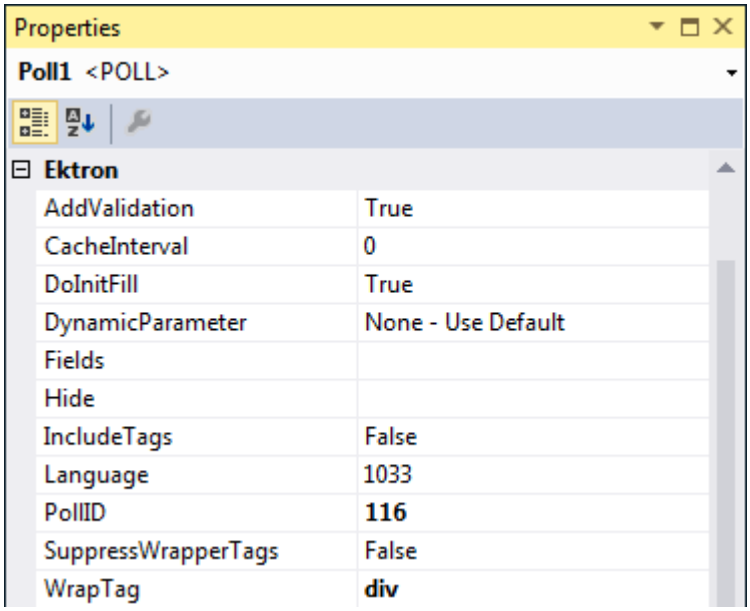
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **Poll** server control and drop it into the desired location on the page.

NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:Poll ID="Poll1" runat="server" />
```

4. Click on `Poll` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



Poll properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AddValidation** (Boolean)
  The AddValidation property is obsolete and ignored. It has no effect. It is always true.
- **CacheInterval** (Double)
  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)
  To make this form block dynamic, select **ekfrm**. When you do, this server control uses the form block passed as a URL parameter.
- **Fields** (FormFieldCollection)
  Displays a list of fields that are defined in the form. These fields are read only. This is an excellent way of displaying the field names used on the form. With this list of names, you can create events using the fields without having to enter the Workarea to see the names.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **IncludeTags** (Boolean)
  Determines if tags are generated automatically or manually.
  - Let Ektron generate form tags automatically if you are developing a pure .aspx page script. To enable this option, set `IncludeTags` property to **True**.
  - Modify HTML form tags in the .aspx file if you are developing an .aspx page and associated code-behind Web form. To enable this option, set `IncludeTags` property to **False**. Here is the default .NET generated form tag:

    ```
    <form id="Form1" method="post" runat="server">
    ```

    Modify the form tag as indicated in red:

    ```
    <form id="Form1" method="post" runat="server"
      OnSubmit="return EkFmValidate(this);">
    ```
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).

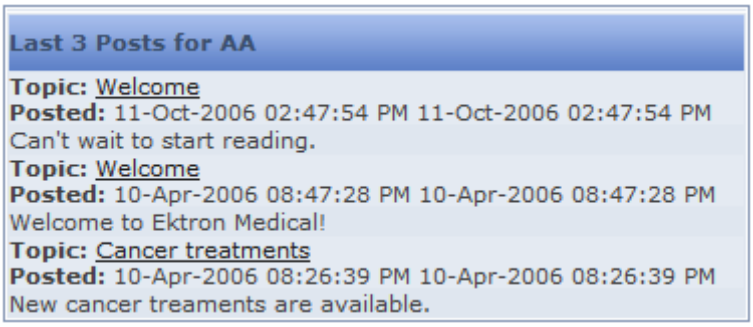
- **PollID** (Long)
  The ID of the poll that appears where you inserted this server control. If you don't know the ID number of the pol, use the CMS Explorer to browse to it.See also: *Browsing your Ektron site using CMS Explorer* on page 2147
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

PostHistory

8.60 and higher

The PostHistory server control displays a list of forum posts for a given user. The posts are displayed in order, by date, and contain the following:

- **Topic**. The topic that contains the post
- **Posted**. The date the post was created
- **Content**. The post's content



Inserting the PostHistory server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

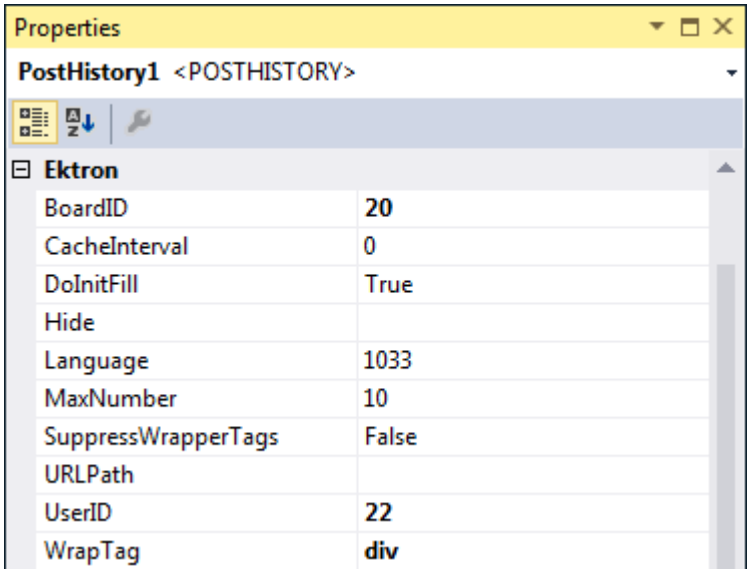
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.

3. Drag the **PostHistory** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:PostHistory ID="PostHistory1" runat="server" />
```

4. Click on `PostHistory` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



PostHistory properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **BoardID** (Long)

  The ID of the discussion board from which to get a user's posts. If you don't know the ID, click **Ellipses** (...), then sign in, browse to and select the discussion board.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **MaxNumber** (Integer)
  The maximum number of posts listed. The default is **10**.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **URLPath** (String)
  The URL path to the page that hosts the Forum server control.
- **UserID** (Long)
  The ID of the user for whom to get the post history.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

RssAggregator

8.50 and higher

An RSS aggregator consumes an RSS feed and displays its information in a readable format. The RssAggregator server control lets you do the same by processing and displaying an RSS feed from any website. This lets you create an information Web page for news, stories, images, lists of music, and so on.

You can create a multi-level information Web page by placing several RssAggregator server controls on a page. For example, create a world news Web page by adding RSS feeds from the NY Times, BBC, and AFP (Agence France-Presse). To add an RSS Aggregator to your website:

1. Drag and drop the RssAggregator server control a Web form.
2. Set the `URL` property to point at the RSS feed. For example, `http://msdn.example.com/rss.xml`

When a site visitor views the Web form, the RSS feed displays properly. If the visitor refreshes the page, information updated by the RSS feed provider appears.

To change an RSS feed's appearance, create a custom XSLT.

Inserting the RssAggregator server control onto a page

PREREQUISITE

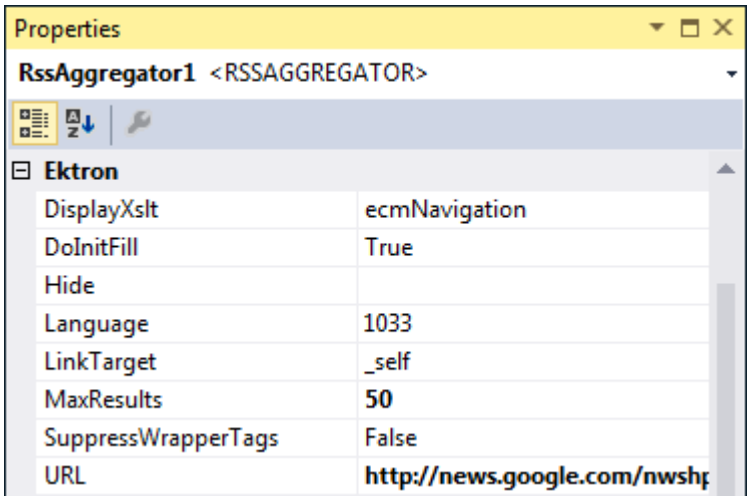
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **RssAggregator** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:RssAggregator ID="RssAggregator1" runat="server" />
   ```

4. Click on `RssAggregator` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



RSS Aggregator properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **DisplayXslt** (String)
  Determines how the information on the page appears.

- **None**. Databind only.
- **ecmNavigation**. Lists the title of every RSS feed item.
- **ecmTeaser**. Lists a title and a description of every RSS feed item.
- **Path to Custom Xslt**. If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file, create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)

  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.

- **Hide** (Boolean)

  Hides or displays the output of the control in design time and run time.

  - **True**. Hide the control output.
  - **False**. Display the control output.

- **Language** (Integer)

  Set a language for viewing the RssAggregator. This property shows results in design-time (in Visual Studio) and at run-time (in a browser).

- **LinkTarget** (ItemLinkTarget)

  Determines the type of window that appears when you click a link in the server control.

  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.

- **MaxResults** (Integer)

  The Maximum number of items from an RSS feed that are returned. 0 (zero) = unlimited.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

- **URL** (String)

  The RSS feed path for the server control. For example:
  `http://msdn.example.com/rss.xml`

Retrieving the XML structure of an RssAggregator

Retrieving the XML structure of XML content allows for greater control over developing XSLs. To retrieve the XML structure:

1. Open a new Web form.
2. Drag and drop a RssAggregator server control onto it.
3. Set the `URL` property.
4. Drag and drop a textbox on the Web form.
5. Set the `TextMode` property to **MultiLine**.

   NOTE: You should set the text box width to at least 400px.

6. Add the following line on the code-behind page.

   ```
   Textbox1.Text. RssAggregator1.XmlDoc.InnerXml
   ```

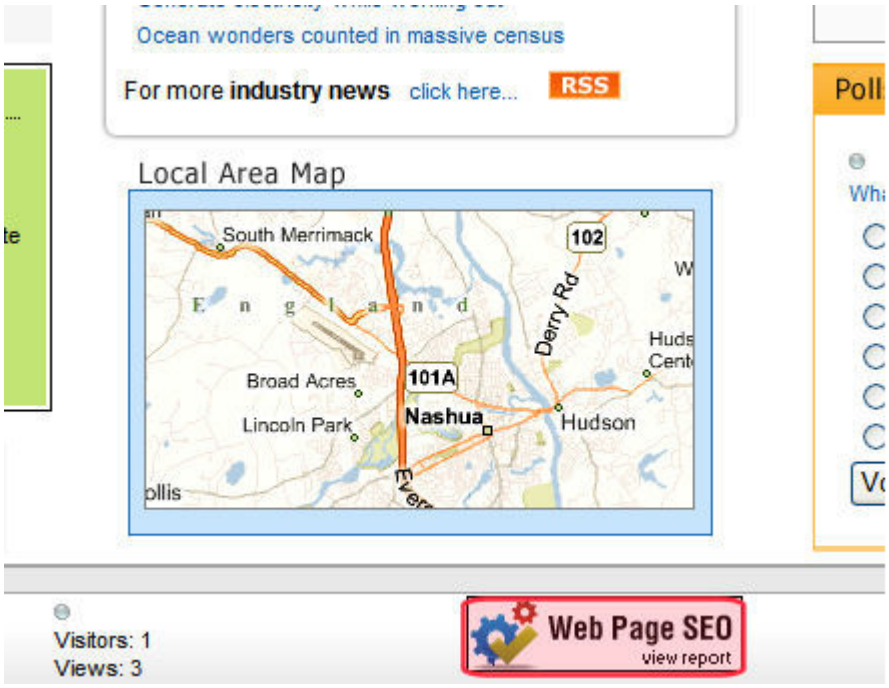
7. Build the project.
8. View the Web form in a browser. The XML structure of the RssAggregator Menu appears in the textbox.

SEO (search engine optimization)

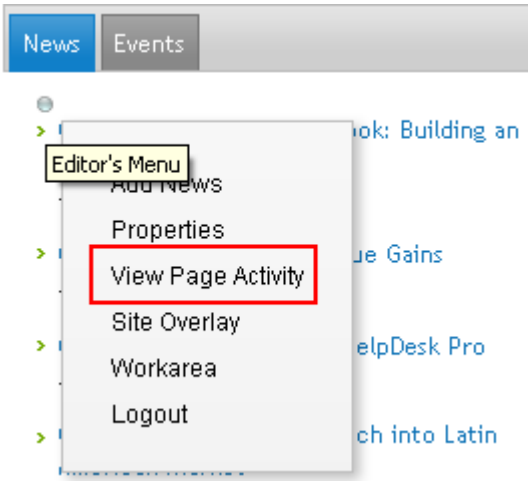
8.60 and higher

The SEO server control analyzes your website for W3C compliance, what information Google has about the page, Alexa rankings, image alt text, keyword density, and metadata. SEO lets you know how (and if) you have set these values.

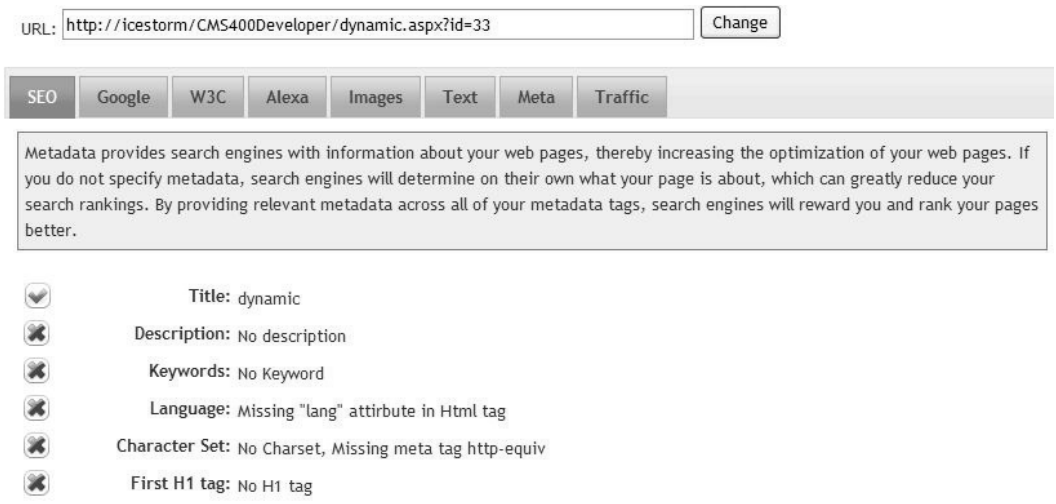
The Search Engine Optimization (SEO) control appears as an image on any template or master page on which you drop it.



You also can access the SEO report by clicking the Web Site Content menu then choosing **View Page Activity**.

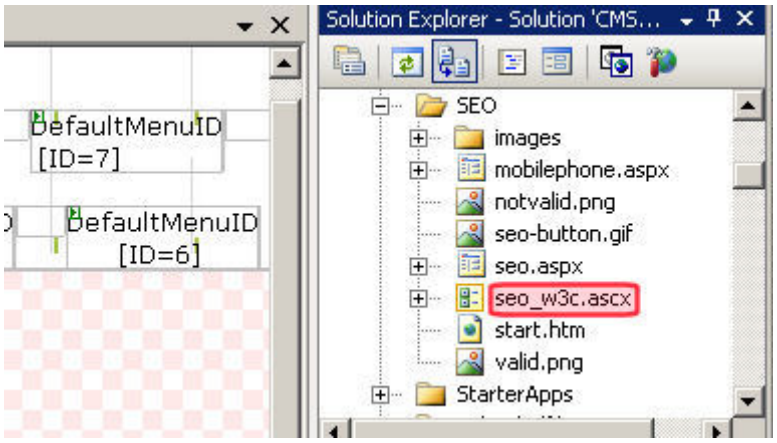


When you are logged in and click the image, you see a detailed report of how search engines evaluate the page. For example, the report runs the page through a W3C validation site, or displays how search engines evaluate the page's text.



Inserting SEO in a website

1. Within Visual Studio, open your website.
2. Open a master page or template on which to place the SEO control.
3. Open Solution Explorer, and your website within it.
4. Open the Workarea then the SEO folder.
5. Drag the SEO control (`seo_w3c.ascx`) onto an area of the page.



6. Save the page.

Viewing a Web page's SEO report

Only logged-in Ektron or Membership users can see the SEO control.The following list summarizes the information on the SEO Report.

- **SEO**. Compares viewed page against a basic SEO checklist, such as if keywords are included, language and charset values, and so on.
- **Google**. Information that search engines have about your page, including pages that link to it, indexed pages on the site and what the page looks like on a mobile device.

- **W3C**. Validates Web page for markup and CSS compliance, and also checks for broken links and the ability of the page to be displayed on mobile devices. By having correct markup that conforms to HTML or XHTML standards, search engines will detect words and phrases correctly, a key element in SEO. Correct markup also ensures that search engine spiders can easily understand the content and navigation on the page. Proper code will make it easier for spiders to parse as expected and digest the content for organic search engines.
- **Alexa**. Alexa is a leader in providing insight on the overall ranking of your website on the Internet. Alexa overview, traffic, related and linked-to sites.
- **Images**. To ensure SEO and compliance, all images on your pages should contain alt tags. This enables searching of the images (since engines cannot read the contents of an image). In addition, alt tags must be used for 508 compliance. This tab identifies all images and their alt tags.
- **Text**. A keyword density analysis report displays text graphically in much the same way that organic search engines see your page. Web pages that are content rich typically yield higher search engine rankings than pages that are mainly images or multimedia based. This report displays words, two-word phrases and 3-word phrases in a cloud format, revealing words and phrases that show up most often on the page. This information helps you to optimize your content and reach your target audience.

- **Meta**. The page's metadata
- **Misc**. Additional search information

SiteMap

8.50 and higher

The Sitemap server control utilizes the folder breadcrumb information in the Workarea to display a sitemap of your site. By choosing the starting point of the sitemap, the max levels to display and applying a style class, you can customize the sitemap. The sitemap appears as indented list when viewed on a Web page.

The contents of the sitemap are defined on the **Breadcrumb** tab, located in the Workarea folder properties. See also: *BreadCrumb* on page 2196.

This section also contains the following topics.

Inserting the SiteMap server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

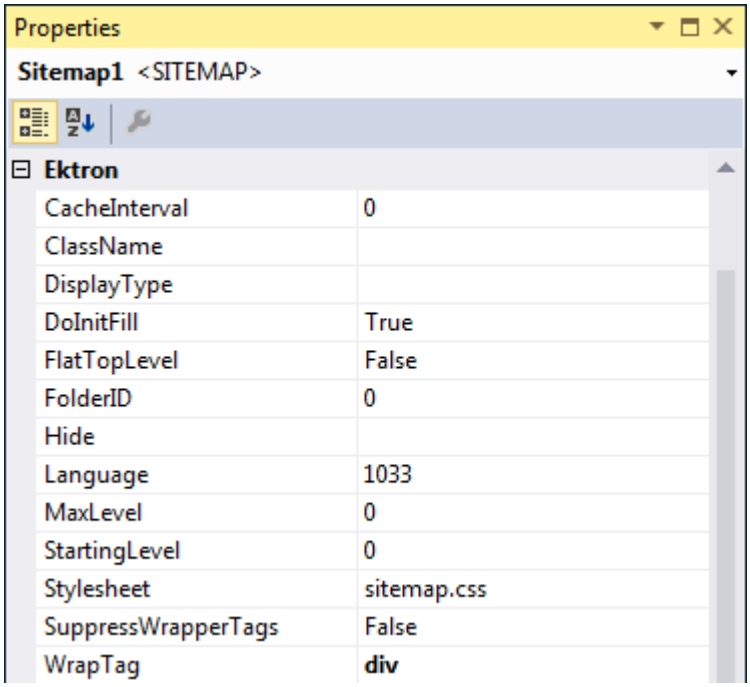
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **SiteMap** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:SiteMap ID="SiteMap1" runat="server" />
   ```

4. Click on `SiteMap` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the

property values.



Sitemap properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **CacheInterval** (Double)

  The number of seconds that a server control's data is cached. The default is 0 (zero). For example, if you want to cache the data for 5 minutes, set to 300. See also: *Caching with server controls* on page 2158.

- **ClassName** (String)

  The style sheet class name used to format the HTML. Leave blank to use the default. To use a new class, add it to `siteroot\Workarea\csslib\sitemap.css`. Then, add the class name to the property.

  IMPORTANT: Files stored in the *siteroot*`\Workarea` folder are overwritten (or deleted) when you upgrade. To avoid problems, back up the files to a folder outside the `siteroot\Workarea` folder.

- **DisplayXslt** (String) (Code-behind only)

  If desired, enter a relative or absolute path to an Xslt that determines the display of the page.

  WARNING! Files stored in the `siteroot\Workarea` folder are overwritten (or deleted) when you upgrade Ektron. To avoid problems, copy the default file to a folder outside the `siteroot\workarea` folder then edit it. If there is no default file,

create the file outside the `siteroot\workarea` folder. Next, in this property, enter the path to that file relative to the site root folder.

- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **FlatTopLevel** (Boolean)
  When set to **True**, include site nodes from the parent folder but not their items. Set to **False** to include all nodes and items.
- **FolderID** (Long)
  The folder ID for the starting point of the site map. To choose the root folder, enter 0 (zero).
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **MaxLevel** (Integer)
  Set the maximum amount of levels to show. 0 (zero) = unlimited.
- **StartingLevel** (Integer)
  Set to the starting level of the site map. If set to 0 (zero), starts from the root.
- **Stylesheet** (String)
  Specify the path to a style sheet for use with this server control. Leave blank to use the default style sheet. To use a custom style sheet, place it in a folder outside the `siteroot\workarea` folder then edit it. Next, in this property, enter the path to the custom style sheet relative to the site root folder.
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Using Sitemap

NOTE: Make sure you have added the sitemap information to your folders' Breadcrumb tab in the Workarea.

To use the Sitemap server control:

1. Open a Web form for which you want to create a sitemap.
2. Drag and drop the Sitemap server control onto an appropriate location of the Web form.
3. Add the starting folder's ID to the `FolderID` property.
4. Set other properties. See also: *Sitemap properties* on page 2523.
5. Save the Web form.
6. Open a browser.
7. View a Web page with the Sitemap server control in it. The sitemap now appears on your site.

Retrieving the XML structure of a site map

Retrieving the XML structure of XML content allows for greater control over developing XSLs. The following is an example of how to retrieve the XML structure:

1. Open a new Web form.
2. Drag and drop a SiteMap server control onto it.
3. Set the `FolderID` property.
4. Drag and drop a textbox on the Web form.
5. Set the `TextMode` property to **MultiLine**.

   NOTE: It is also recommended that you set the width of the text box to at least 400px.

6. On the code-behind page, add the following line.
7. `Textbox1.Text = SiteMap1.XmlDoc.InnerXml`
8. Build the project.

9. View the Web form in a browser. The XML structure of the Site Map appears in the text box.

WebCalendar

8.50 and higher

The WebCalendar server control displays Ektron calendars. See also: Working with Calendars.

NOTE: On a PageBuilder page, use the Calendar widget to insert a calendar.

The following code example shows the WebCalendar server control.

```
<cms:WebCalendar ID="webcalendar" runat="server"
 DynamicParameter="calendar_id" DisplayType="Day">
     <DataSource>
   <cms:CalendarDataSource defaultId="724"sourceType="SystemCalendar"/>
 </DataSource>
</cms:WebCalendar>
```

The following list shows properties set inside the `<DataSource> <CalendarDataSource>` tag.

- **backColor** (String)
  The background color of the calendar event. If you set this value to `AutoSelect`, the next available color in the list is chosen automatically.
- **defaultId** (String)
  The Id of the SystemCalendar, GroupCalendar or UserCalendar to display on the Web page. For more information about using this property, see Combining Web Calendars (Mashups).
- **queryParam** (String)
  The parameter that may be in the query string to mash-up additional calendars. This can be any Calendar SourceType. For example, if the value is set to **uid,** then the querystring can be:

  `.../calendar.aspx?calendar_id=”55”&uid=”440”`

  For more information about using this property, see Combining Web Calendars (Mashups).
- **sourceType** (String)
  Choices are **SystemCalendar, GroupCalendar, UserCalendar**. See also: Combining Web Calendars (Mashups)

  NOTE: The CategoryID property set inside the `<DataSource> <CategoryID>` tag is used to identify taxonomy categories to filter calendar events.

Inserting the WebCalendar server control onto a page

PREREQUISITE

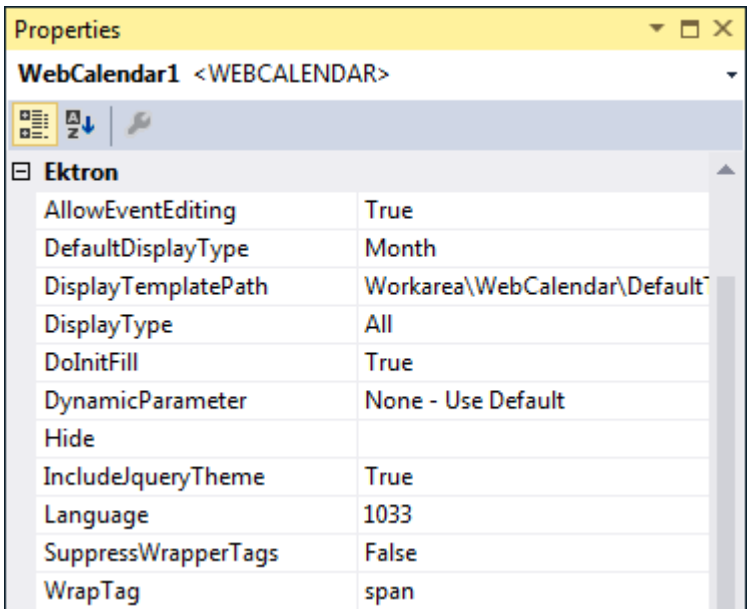
You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **WebCalendar** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

```
<CMS:WebCalendar ID="WebCalendar1" runat="server"></CMS:WebCalendar>
```

4. Click on `WebCalendar` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify the property values.



WebCalendar properties

The following are Ektron-specific server control properties. For information about native .NET properties such as font, height, width and border style, use Visual Studio® help.

- **AllowEventEditing** (Boolean)

  Determines whether users can add or edit calendar events. Default is **True**.

- **DefaultDisplayType** (String)

  Specify the default calendar display. The default is **Month**, which means that the month view appears on the Web page. The site visitor can change the view.

- **DisplayTemplatePath** (String)

  Specify the path to the code that controls how events are displayed. Default is blank.

- **DisplayType** (String)
  Determines which time periods appear on the calendar. Choices are **All**, **Day**, **Month**, **Week**. Default is **All**.
- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicParameter** (String)
  To make this calendar dynamic, select **calendar_id**. When you do, this server control uses the calendar passed as a URL parameter. To exclude this function, choose **None- Use Default**. Only Calendars with `sourcetype=SystemCalendar` may be used in the querystring for this value. For example, where `DyanamicParameter="calendar_id"`, the URL may read `http://mysite.com/calendar.aspx?calendar_id="55"`. In this case, calendar 55 must be of the SystemCalendar type.
- **Hide** (Boolean)
  Hides or displays the output of the control in design time and run time.
  - **True**. Hide the control output.
  - **False**. Display the control output.
- **Language** (Integer)
  Set a language for viewing content; shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **SuppressWrapperTags** (Boolean)
  Suppresses the output of the span/div tags around the control.
  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

WebSearch

IMPORTANT: Deprecated. The WebSearch server control is deprecated as of Release 8.50. Instead of the WebSearch control, you should convert to using templated search. If you are already using the WebSearch server control, you can continue to do so with one important exception: the Solr search provider is incompatible with the WebSearch server control.

The Web Search server control lets you customize the behavior of search. You place this control on any Web form from which a site visitor can search your site. For more information about the site visitor experience, see Setting Up Search for Your Website.

For new pages, developers are encouraged to replace the Web Search server control with Templated Search server controls. See also: *Templated server controls* on page 2045.

Inserting the WebSearch server control onto a page

PREREQUISITE

You must have installed the server controls. See *Installing server controls into Visual Studio Toolbox* on page 2135.

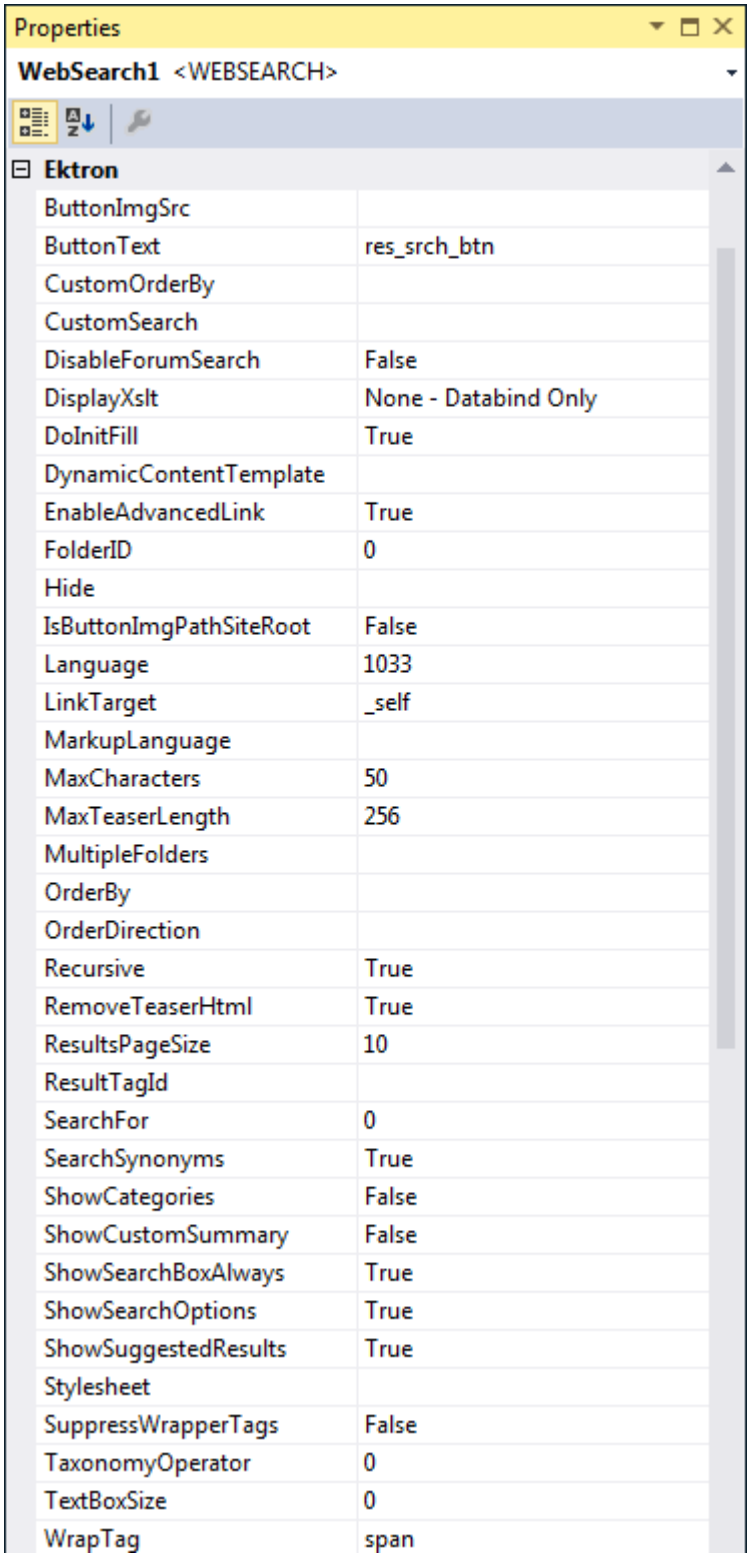
1. In Visual Studio, choose **View > Toolbox**.
2. Click the **Ektron server control** tab to display the server controls.
3. Drag the **WebSearch** server control and drop it into the desired location on the page.

   NOTE: You also can place the cursor on the page where you want the server control, then double click the server control that you want.

   ```
   <CMS:WebSearch ID="WebSearch1" runat="server" />
   ```

4. Click on `WebSearch` in the code to display and modify the control's properties using the Properties window of Visual Studio. The page is updated as you modify

the property values.



WebSearch properties

The following table describes the Web Search server control properties.

- **ButtonImgSrc**(String)

  If you want to display an image on the submit button, enter the server path to the image.

- **ButtonText**(String)

  The button text if no image source is identified in the `ButtonImgSrc` property. If an image source is identified, this is alternate text for the button.

- **CustomOrderBy**(String)

  Provide a property's Friendly Name defined in the Indexing Service to sort search results by that property. For example, if you define DocAuthor, results are sorted by the document's author. A property's Friendly Name can be found in **Computer Management > Services and Applications > Indexing Service > Your Index > Properties > Friendly Name** column.

  Results can be ascending or descending based on `OrderDirection`. If you enter an invalid property, no search results are returned.

  If you specify `CustomOrderBy` and `OrderBy`, the `OrderBy` property is ignored.

- **CustomSearch**(String)

  If you want the search to include folders outside of Ektron, enter the folder names here. Separate multiple items with a comma. You do not need to enter the folder path, but it must reside within the site root folder.

- **DisableForumSearch**(Boolean)

  Set to true if you want to remove Forums from the list of content types that appears on the Search server control. The default value is false. Regardless of this setting, if a user selects **Site** from the list, forum posts are searched.

- **DisplayXslt**(String)

  Determines the display of the search results page.

  - **None**. Databind only
  - **ecmNavigation**. Lists the title of every content item found by the search
  - **ecmTeaser**. Lists the title and summary of every content item found by the search
  - **ecmUnOrderedList**. Sorts in no particular order. Shows the title and content summary.
  - **Path to Custom Xslt**. Enter the path to an Xslt that determines the display of the page.

    WARNING! If you specify an external Xslt file, it is strongly recommended that you do not store this file in your site's Workarea folder. If you store this file in the Workarea folder, the file will be lost when you upgrade.

    WARNING! If you enter a valid EkML file at the MarkupLanguage property, this property value is ignored.

- **DoInitFill** (Boolean)
  By default, Fill occurs during the Page_Init event. Set to **false** if you want to postpone the fill-action until later. In this case, Fill is automatically called during the Page Render event. You might do this if you need to set or change a property on the control in code-behind and have it render with your changes shown.
- **DynamicContentTemplate**(String)
  Sets the template for dynamic content. This property overrides any quicklink template for the content.
- **EnableAdvancedLink** (Boolean)
  Set to **true** to display an additional tab (**Advanced**) on the Search control.
- **FolderID**(String)
  The folder at which the search begins. The starting folder need not be the root folder. The `Recursive` property determines if the search examines this folder's subfolders.
- **Hide** (Boolean)
  Select **False** to display this server control on the page. Select **True** to suppress it.
- **IsButtonImagePathSiteRoot** (Boolean)
- **Language** (Integer)
  If the template on which this server control resides includes a language selection control, and you want to let the site visitor select the language, enter zero (0). Otherwise, click the field, then the ellipsis button and a popup box appears. Select a language from the list.

  This property shows results in design-time (in Visual Studio) and at run-time (in a browser).
- **LinkTarget** (String)
  Determines the type of window that appears when you click a link in the server control.
  - **_Self** (default). Opens in same window.
  - **_Top**. Opens in parent window.
  - **_Blank**. Opens in new window.
  - **_Parent**. Opens in the parent frame.
- **MarkupLanguage** (String)
  Enter the template markup file (`.ekml`) that controls the display of this server control. To use the default .ekml file, leave this field blank.

  The default file follows this pattern: `\siteroot\Workarea\template\this object\objectname.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.

To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in this property, enter the path to that file relative to the site root folder.

See also: *Ektron Markup Language* on page 2633

NOTE: If you enter a valid EkML file at the **MarkupLanguage** property, the **Displayxslt** property value is ignored. If the EkML file contains the `[$ImageIcon]` variable, the `IncludeIcons` property acts as True.

- **MaxCharacters** (Integer)

  The maximum number of characters the Search text box accepts. If you enter less than 50, set the `TextBoxSize` property to the same number.

- **MaxTeaserLength** (Integer)

  Limits the length of any returned content's abstract. To allow unlimited length, set to zero (0). This property is active only if both of these conditions are true.

  - You use the `DisplayXslt` property to identify an xslt and ecmteaser as a value of that property. If you enter an .ekml file at the MarkupLanguage property, this value is ignored.
  - the `ShowCustomSummary` property is set to `false`. If it is set to `true`, the entire summary appears in search results.

- **OrderBy** (String)

  The field that determines the sorting of search results.

  NOTE: The `Order Direction` field determines the direction of the search results. For example, if you sort by ID and `Order Direction` is set to **Descending**, the results sort by ContentID with the highest ID number at the top of the list.

  - Title. The content title (alphabetical)
  - ID. The content ID number
  - Date Created. The date the content was created
  - Date Modified. The date the content was most recently modified
  - Editor. The user who last edited the content (alphabetical)
  - Rank. The content's rank.

- **OrderDirection** (String)

  The direction in which search results are sorted. The default is Ascending.

  - **Ascending**. Alphabetical results from A to Z; numeric values low to high; dates from oldest to most recent
  - **Descending**. Alphabetical results from Z to A; numeric values high to low; dates from most recent to oldest

- **Recursive** (Boolean)

  Determines whether to search sub-folders of the starting folder. The `FolderID` property determines the starting folder.

- **RemoveTeaserHTML** (Boolean)
  Set to true if you want to remove HTML tags from the content summary within search results.
- **ResultsPageSize** (Integer)
  Set the maximum number of search results on a page. If a search returns more than this many results, the following text appears below the last one:

  Result Page: 1 2 3 Next

  The user can click **Next** or a number to view additional results.

  This property defaults to the value set at the `ek_PageSize` element in the *siteroot*\`web.config` file.

  Property's Effect on Suggested Results

  Only the number of Suggested Results up to this maximum appear. If more than this number should display, they do not. This is unlike natural search results, whose additional links are available via numbers below the maximum page size.
- **ResultTagId** (String)
  Lets you designate where search results appear. You can place search criteria in one area of a Web form and results in another. For example, you have the following tag.

  ```
  <span id="results"></span>
  ```

  In this case, enter **results** for this property's value.
- **SearchFor** (String)
  Choose the type of content that may be searched via this control.
  - All
  - HTML
  - Documents
  - Images
  - Multimedia
  - Discussion Forums
  - Tags
  - eCommerce products
  - PageBuilder

  If the value is anything other than **All**, this server control only examines the selected content type.

  IMPORTANT: If this property is set to anything other than **All**, the search options drop-down does not appear.
- **SearchSynonyms** (Boolean)
  If set to true, the Synonym Search is included with the search. If false, Synonym Sets are ignored.

- **ShowCategories** (Boolean)

  If set to true, this server control displays a **Filter by Category** option, which helps a site visitor zero in on relevant content. If false, **Filter by Category** does not appear.

  IMPORTANT: In order for the **Filter by Category** option to appear, the `ShowSearchBoxAlways` property must be set to true.

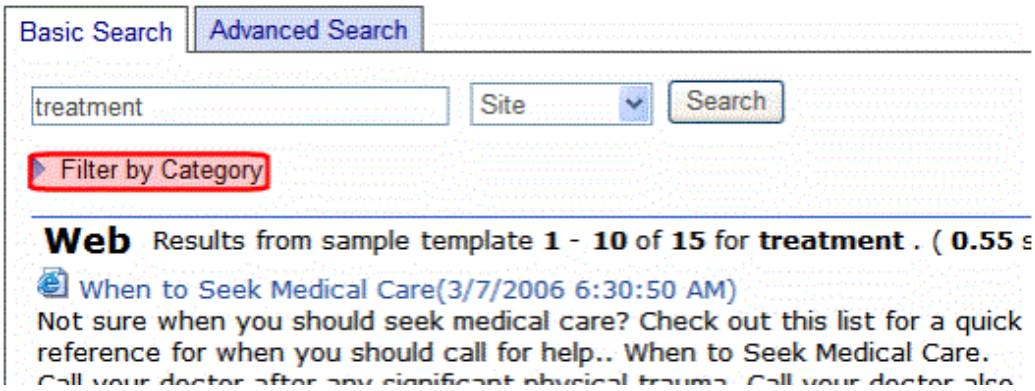
  The Taxonomy feature lets users assign information categories to content. For example, if your organization is a university, taxonomy categories might be Athletics, Alumni, Admissions, Academic Departments, and so on. As new content is created, users should apply relevant taxonomy categories to it. This enables a site visitor to search by category *and* search terms. For example, if the search term is calendar and the category is Athletics, the search would typically return calendars of sports teams but not other calendars, such as those for graduation, exams, or parent weekend.

  NOTE: This property depends on the assignment of taxonomy categories to content. If they are not, the filter hides relevant but unclassified content.
  For example, an author creates an article on "Treating Heart Disease" but doesn't assign a taxonomy category to it. If a site visitor on a search page selects **Filter by Category** then the **Medical Forum > Heart Disease** category, he will not find that article.

  A developer can control whether results must match *all* categories selected in the **Filter by Category** tree, or *at least one* category. To display results that must match all categories, set the `TaxonomyOperator` property to `And`. To show results that match one or more categories, set `TaxonomyOperator` to `Or`. By default, the property is set to `Or`, which provides a wider range of results.

  Effect of Setting ShowCategories to True

  If you set `ShowCategories` to true, initially the site visitor sees no difference. Upon entering a search term that exists in content to which a taxonomy category is assigned and clicking **Search**, **Filter by Category** appears above the results (illustrated below).



- **ShowCustomSummary** (Boolean)

  If set to **true**, the search results display the content item's summary. If false, the search results display the characterization. The default is false.

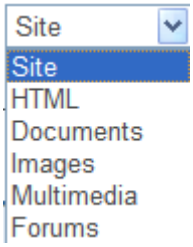
NOTE: If this property is set to true, the `MaxTeaserLength` property is ignored. So, the entire summary appears with search results, regardless of length.

- **ShowSearchBoxAlways** (Boolean)

  If set to true, the search box appears on the PostBack screen. If false, the search box does not appear on the PostBack screen. The default is true.

- **ShowSearchOptions** (Boolean)

  If set to true, the following drop-down appears to the right of the Search box.



  A site visitor can click an option to limit the search by content type. If the user accepts the default value, **Site**, all content types are searched.

WARNING! If the `SearchFor` property is set to anything other than **All**, the search options drop-down does not appear.

NOTE: If the `DisableForumSearch` property is set to true, **Forums** does not appear in the drop-down.

- **ShowSuggestedResults** (Boolean)

  If set to `true`, Suggested Results related to the search term appear. If `false`, Suggested Results do not appear.

NOTE: If the `ResultsPageSize` property value is less than the number of suggested results applied to a term or synonym set, only the property's number of results appears. For example, if you assign five links to a Suggested Result set but set `ResultsPageSize` to three, only the first three results appear.

- **Stylesheet** (String)

  If you want to define a style sheet for the search results page, specify its path relative to the site root folder. For example: `Workarea\csslib\mytest.css`. Leave blank to use the default style sheet, `webroot\Workarea\csslib\search.css`.

WARNING! If you enter a valid EkML file at the MarkupLanguage property, or a value at the DisplayXslt property, this property is ignored.

- **SuppressWrapperTags** (Boolean)

  Suppresses the output of the span/div tags around the control.

  - **True**. Suppress wrap tags.
  - **False** (default). Allow wrap tags.

- **TaxonomyOperator** (Enum-TaxCategoryOperator)
  Select whether to use an `And` or `Or` operator when filtering results by taxonomy. The default value is `Or`.
  - **And**. Only results that match all categories selected in the Filter by Category tree appear. For example, if a site visitor is searching for a medical document in the Hospital and Doctor's Office categories, only documents assigned to both categories appear.
  - **Or**. If more than one category is selected in the Filter by Category tree, results needs to only match one category to be shown.

  NOTE: For this property to be active, the `ShowCategories` property must be set to `true`.

- **TextBoxSize** (Integer)
  The size of the search text box for user input, in number of characters.
- **WrapTag** (String)
  Lets a developer specify a server control's tag.
  - **Span** (default). Designate an inline portion of an HTML document as a span element.
  - **Div**. Apply attributes to a block of code.
  - **Custom**. Lets you use a custom tag.

Property usage table

Web Search server control properties generally affect the control in 1 of 3 ways.

- **Search Display**. The server control's appearance on a Web page
  - ButtonImgSrc
  - ButtonText
  - DisableForumSearch
  - EnableAdvanced Link
  - Hide
  - Language
  - MaxCharacters
  - ShowSearchOptions
  - Text Box Size
  - WrapTag
- **Search Criteria**
  - CustomSearch
  - FolderID
  - Language
  - Recursive
  - SearchFor

- ShowCategories
- ShowSearchOptions
- ShowSuggested Results

- **Search Results Display**
  - CustomOrderBy
  - DynamicContentTemplate
  - DisplayXslt
  - Hide
  - Language
  - LinkTarget
  - MaxTeaserLength
  - OrderBy
  - OrderDirection
  - RemoveTeaserHtml
  - ResultsPageSize
  - ResultTagId
  - ShowCustomSummary
  - ShowSearchBoxAlways

IQueryable data sources for Ektron data types

9.20 and higher

This feature provides an IQueryable interface to interact with most core Ektron data types. *Linq support data classes* below provides a full list of supported data types and properties supported for querying.

The feature lets developers greatly simplify Framework API queries using expression trees or LINQ. For example:

```
var dataset = Ektron.Cms.Linq.EktronContext<Ektron.Cms.ContentData>.Source
                .Where(item => item.DateModified < DateTime.Now.AddMonths(-1) && item.FolderId == 30)
                .OrderBy(item => item.Title)
                .Skip(10)
                .Take(10);
```

Or the functionally equivalent:

```
var dataset2 = (from item in
Ektron.Cms.Linq.EktronContext<Ektron.Cms.ContentData>.Source
                where item.DateModified < DateTime.Now.AddMonths(-1) && item.FolderId == 30
                orderby item.Title
                select item);
```

While these calls are simpler to write, they still use the framework API to retrieve items underneath. So, they adhere to internal business logic, such as retrieving items to which the current user has permission. The calls also function with all predefined unity containers, such as WCF or Cache.

Available operators.

- Where - Within a **Where** predicate, the following operators are supported.
  - !
  - ==
  - !=
  - >
  - >=
  - <
  - <=
  - Contains
  - StartsWith
- OrderBy
- OrderByDescending
- Skip and
- Take

Linq support data classes

Linq support has the following data classes.

ActivityCommentData

PropertyName	WhereSupport	SortSupport
ActivityId	yes	yes
Comment	yes	yes
DateCreated	yes	yes
Id	yes	yes
UserAvatar	yes	yes
UserDisplayName	yes	yes
UserId	yes	yes

ActivityData

PropertyName	WhereSupport	SortSupport
ActivityTypeId	yes	yes
CommentCount	yes	yes
Date	yes	yes
Id	yes	yes
LanguageId	yes	yes
Message	yes	yes
ObjectId	yes	yes
ScopeObjectId	yes	yes
SiteId	yes	yes
UniqueId	yes	yes

ActivityTypeData

PropertyName	WhereSupport	SortSupport
ActionType	yes	yes
DisplayInPreferences	yes	yes
Id	yes	yes
IsPublishByDefault	yes	yes
Name	yes	yes
ObjectType	yes	yes
Scope	yes	yes
SupportsComments	yes	yes

AddressData

PropertyName	WhereSupport	SortSupport
City	yes	yes
Company	yes	yes
Id	yes	yes
Name	yes	yes
PostalCode	yes	yes

AliasData

PropertyName	WhereSupport	SortSupport
Alias	yes	yes
ConfigurationId	yes	yes

Id	yes	yes
IsDefault	yes	yes
IsEnabled	yes	yes
LanguageId	yes	yes
NodeId	yes	yes
QueryString	yes	yes
QueryStringAction	yes	yes
SiteId	yes	yes
TargetId	yes	yes
TargetURL	yes	yes
Type	yes	yes

AliasRuleData

PropertyName	WhereSupport	SortSupport
AliasPriority	yes	yes
ExcludedPathId	yes	yes
FileExtensionId	yes	yes
Id	yes	yes
IsDeleted	yes	yes
IsEnabled	yes	yes
LanguageId	yes	yes
Name	yes	yes

PageTypeId	yes	yes
Prefix	yes	yes
SiteId	yes	yes
SourceId	yes	yes
SourceParamName	yes	yes
TargetTemplateSource	yes	yes
Type	yes	yes

CmsDeviceConfigurationData

PropertyName	WhereSupport	SortSupport
DeviceType	yes	yes
DisplayFor	yes	yes
Height	yes	yes
Id	yes	yes
Name	yes	yes
Order	yes	yes
Width	yes	yes

CmsMessageData

PropertyName	WhereSupport	SortSupport
HtmlBody	yes	no
Id	yes	yes
IsDefaultMessage	yes	yes

LanguageId	yes	yes
SiteId	yes	yes
Subject	yes	yes
TextBody	yes	no
Title	yes	yes

CmsMessageTypeData

PropertyName	WhereSupport	SortSupport
Id	yes	yes
Name	yes	yes
Scope	yes	yes

CommerceAuditData

PropertyName	WhereSupport	SortSupport
DateCreated	yes	yes
FormattedMessage	yes	no
Id	yes	yes
IPAddress	yes	yes
Message	yes	yes
OrderId	yes	yes
Title	yes	yes
UserId	yes	yes

CommunityGroupData

PropertyName	WhereSupport	SortSupport
AllowMembersToManageFolders	yes	yes
CreatedDate	yes	yes
CreatedUser	yes	yes
EnableCalendar	yes	yes
EnableDiscussionBoard	yes	yes
EnableDistributeToSite	yes	yes
EnableDocumentsInNotifications	yes	yes
EnableGroupEmail	yes	yes
EnableTodoList	yes	yes
GroupEnroll	yes	yes
GroupEnroll	yes	yes
GroupId	yes	yes
GroupId	yes	yes
GroupImage	yes	yes
GroupImage	yes	yes
GroupLevel	yes	yes
GroupLevel	yes	yes
GroupLocation	yes	yes
GroupLocation	yes	yes

GroupName	yes	yes
GroupName	yes	yes
GroupParentId	yes	yes
GroupParentId	yes	yes
GroupPath	yes	yes
GroupPath	yes	yes
LongDescription	yes	yes
ShortDescription	yes	yes

ContentAssetData

PropertyName	WhereSupport	SortSupport
AssetId	yes	yes
DateCreated	yes	yes
DateModified	yes	yes
EditorFirstName	yes	yes
EditorLastName	yes	yes
FolderId	yes	yes
FolderName	yes	yes
Id	yes	yes
IsPublished	yes	yes
LanguageId	yes	yes
Status	yes	yes

SubType	yes	yes
Title	yes	yes
Type	yes	yes
UserId	yes	yes

ContentCollectionData

PropertyName	WhereSupport	SortSupport
DateCreated	yes	yes
Description	yes	yes
FolderId	yes	yes
Id	yes	yes
IsApprovalRequired	yes	yes
LastEditDate	yes	yes
Status	yes	yes
Template	yes	yes
Title	yes	yes
UserId	yes	yes

ContentData

PropertyName	WhereSupport	SortSupport
ContentPath	yes	yes
DateCreated	yes	yes
DateModified	yes	yes

EditorFirstName	yes	yes
EditorLastName	yes	yes
EndDate	yes	yes
EndDateActionType	yes	yes
ExpireDate	yes	yes
ExternalTypeId	yes	yes
FolderId	yes	yes
FolderName	yes	yes
GoLiveDate	yes	yes
Id	yes	yes
IsPublished	yes	yes
IsSearchable	yes	yes
LanguageId	yes	yes
Path	yes	yes
PermissionInheritedFrom	yes	yes
Status	yes	yes
SubType	yes	yes
Title	yes	yes
Type	yes	yes
UserId	yes	yes

ContentMetaData

PropertyName	WhereSupport	SortSupport
Id	yes	yes
Language	yes	yes
Name	yes	yes
ObjectType	yes	yes
Required	yes	yes

ContentRatingData

PropertyName	WhereSupport	SortSupport
ContentId	yes	yes
ContentLanguageId	yes	yes
ContentReviewState	yes	yes
Id	yes	yes
RatingDate	yes	yes
UserId	yes	yes
VisitorId	yes	yes

ContentWorkflowActivityData

PropertyName	WhereSupport	SortSupport
ActivityName	yes	yes
AllowEdit	yes	yes
ApprovalOrder	yes	yes

DefinitionId	yes	yes
EscalationObjectId	yes	yes
EscalationTimeInterval	yes	yes
EscalationUserType	yes	yes
Id	yes	yes
UserId	yes	yes
UserType	yes	yes

ContentWorkflowDefinitionData

PropertyName	WhereSupport	SortSupport
ContentId	yes	yes
FolderId	yes	yes
Id	yes	yes
IsActive	yes	yes
LanguageId	yes	yes
VersionMap	yes	yes

ContentWorkflowInstanceData

PropertyName	WhereSupport	SortSupport
ApproverId	yes	yes
ContentId	yes	yes
CurrentState	yes	yes
DateModified	yes	yes

DateStarted	yes	yes
DefinitionId	yes	yes
Id	yes	yes
InstanceId	yes	yes
IsEscalated	yes	yes
PreviousState	yes	yes
Status	yes	yes

CountryData

PropertyName	WhereSupport	SortSupport
Enabled	yes	yes
Id	yes	yes
LongIsoCode	yes	yes
Name	yes	yes
ShortIsoCode	yes	yes

CouponData

PropertyName	WhereSupport	SortSupport
AppliesToQuantity	yes	yes
Code	yes	yes
CouponType	yes	yes
CurrencyId	yes	yes
Description	yes	yes

DiscountType	yes	yes
DiscountValue	yes	yes
ExpirationDate	yes	yes
Id	yes	yes
IsActive	yes	yes
IsApplicabletoSubscriptions	yes	yes
IsCombinable	yes	yes
IsRedeemable	yes	yes
MaximumAmount	yes	yes
MaximumUses	yes	yes
MinimumAmount	yes	yes
OnePerCustomer	yes	yes
StartDate	yes	yes
UseCount	yes	yes

CreditCardTypeData

PropertyName	WhereSupport	SortSupport
Id	yes	yes
IsAccepted	yes	yes
Name	yes	yes

CurrencyData

PropertyName	WhereSupport	SortSupport

AlphaIsoCode	yes	yes
CultureCode	yes	yes
Enabled	yes	yes
Id	yes	yes
Name	yes	yes

CustomerData

PropertyName	WhereSupport	SortSupport
BillingAddressId	yes	yes
DateCreated	yes	yes
DisplayName	yes	yes
EmailAddress	yes	yes
FirstName	yes	yes
Id	yes	yes
IsDeleted	yes	yes
IsMembershipUser	yes	yes
LanguageId	yes	yes
LastName	yes	yes
ShippingAddressId	yes	yes
TotalOrders	yes	yes
TotalOrderValue	yes	yes
UserName	yes	yes

DeviceBreakpointData

PropertyName	WhereSupport	SortSupport
Description	yes	yes
FileLabel	yes	yes
Height	yes	yes
Id	yes	yes
Name	yes	yes
Width	yes	yes

DxHMappingData

PropertyName	WhereSupport	SortSupport
Adapter	yes	yes
Connection	yes	yes
DateCreated	yes	yes
Id	yes	yes
MappingTaskId	yes	yes
SourceObjectDefinition	yes	yes
SourceObjectDefinitionId	yes	yes
TargetObjectDefinition	yes	yes
TargetObjectDefinitionId	yes	yes
Title	yes	yes
WorkflowId	yes	yes
WorkflowName	yes	yes

Data class:DxHUserConnectionData

PropertyName	WhereSupport	SortSupport
ConnectionName	yes	yes
ConnectorName	yes	yes
ConnectorObjectName	yes	yes
ExternalUserKey	yes	yes
Id	yes	yes
VisitorId	yes	yes

EntryData

PropertyName	WhereSupport	SortSupport
ContentStatus	yes	yes
CurrencyId	yes	yes
DateModified	yes	yes
EndDate	yes	yes
EntryType	yes	yes
FolderId	yes	yes
GoLive	yes	yes
Html	yes	no
Id	yes	yes
IsArchived	yes	yes
IsBuyable	yes	yes

IsPublished	yes	yes
IsSearchable	yes	yes
LanguageId	yes	yes
LastEditorFirstName	yes	yes
LastEditorLastName	yes	yes
ListPrice	yes	yes
SalePrice	yes	yes
Sku	yes	yes
Status	yes	yes
Summary	yes	no
TaxClassId	yes	yes
Title	yes	yes

FacebookCategoryData

PropertyName	WhereSupport	SortSupport
Code	yes	yes
Id	yes	yes
IsEnabled	yes	yes
Name	yes	yes
SiteId	yes	yes

FavoriteItemData

PropertyName	WhereSupport	SortSupport

Description	yes	no
Id	yes	yes
ObjectId	yes	yes
ObjectType	yes	yes
TaxonomyId	yes	yes
Title	yes	yes
Url	yes	yes
UserId	yes	yes

FlagDefData

PropertyName	WhereSupport	SortSupport
Description	yes	yes
Id	yes	yes
Language	yes	yes
Name	yes	yes

FolderData

PropertyName	WhereSupport	SortSupport
Description	yes	yes
FolderType	yes	yes
Id	yes	yes
IsCommunityFolder	yes	yes
IsDomainFolder	yes	yes

IsHidden	yes	yes
Name	yes	yes
NameWithPath	yes	yes
ParentId	yes	yes
PrivateContent	yes	yes

FormData

5

CMS strategies

IMPORTANT: In versions previous to 8.00, developers used the Plug-in Extension Wizard to extend the system. As of version 9.00, Ektron no longer raises plugin extension events. You *must* use extensible strategies and events to extend the CMS.

Developers can extend or modify the behavior of Ektron with software modules called *strategies*. The benefits of strategies include:

- You have full access to the Web application context, the HTTP context, the session information, data stored in cache, and APIs.
- You have more functionality available (such as the notification system) through API access.
- You can attach and set breakpoints in the same way you debug Web applications.
- Performance is improved because Web services are not required.
- Strategies run in the context of your Web application, so Windows Services are not required.

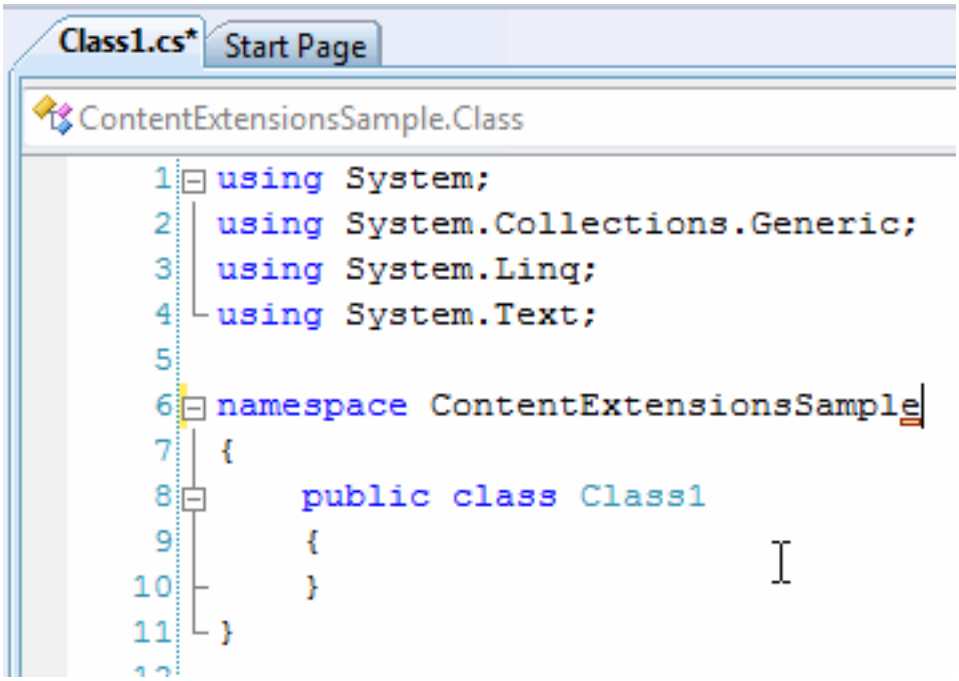
This section also contains the following topics.

Creating a CMS extension with Visual Studio

Creating a CMS extension with Visual Studio

1. In Visual Studio, open your website.
2. Add a new C# class to the `App_Code` directory.

3. The `Class1.cs` code page appears in the editor.



4. Add references to the following assemblies if they are not already added.

```
Ektron.CMS.Common
Ektron.CMS.Contracts
Ektron.CMS.ObjectFactory
```

   If the assemblies are not installed in Visual Studio, you can find them in the *siteroot*`/bin` directory.

5. Add the following `using` statements to the top of the class.

```
using Ektron.Cms;
using Ektron.Cms.Common;
using Ektron.Cms.Extensibility;
using Ektron.Cms.Extensibility.Content;
```

6. Update the class to extend `Ektron.Cms.Extensibility.ContentStrategy`.

```
namespace Cms.Extensions.Samples { public class ContentExtensionsSample :
ContentStrategy { } }
```

   NOTE: The namespace in this code is Cms.Extensions.Samples.

7. Override the method for the event you want to catch; in this example, `override OnBeforeAddContent()`. The code to modify the Title is as follows.

```
public override void
  OnBeforeAddContent(ContentData contentData, CmsEventArgs eventArgs)
  { contentData.Title += " modified"; }
```

The final code is as follows:

```
using System;
using System.Collections.Generic;
using System.Text;
using Ektron.Cms;
using Ektron.Cms.Common;
```

```
using Ektron.Cms.Extensibility;
using Ektron.Cms.Extensibility.Content;
namespace Cms.Extensions.Samples
{
public class ContentExtensionsSample : ContentStrategy
  { public override void
    OnBeforeAddContent(ContentData contentData, CmsEventArgs eventArgs)
    { contentData.Title += " modified"; }
  }
}
```

NOTE: More examples of extensions are found in the Ektron SDK folder. The default location for this folder is `Program Files\Ektron\CMS400SDK\CMS Extensions`. Also, sample eCommerce extensions are found in `Program Files\Ektron\CMS400SDK\Commerce\Events`.

Registering a strategy in the ObjectFactory

After creating an extension, register it in the `<webroot>/ObjectFactory.config` file. This file already contains `GoogleGeoCoder` objectStrategies, so you need to add a new Name in the Content area. In this example, the code looks like this.

```
<objectFactory>
  <objectStrategies>
    <add name="Content">
      <strategies>
        <add name="MyFirstExample"
          type="Cms.Extensions.Samples.ContentExtensionsSample"/>
        <add name="GoogleGeoCoder"
          type="Cms.Extensions.GoogleGeoCoder.ContentStrategy"/>
      </strategies>
    </add>
    <add name="User">
      <strategies>
        <add name="GoogleGeoCoder"
          type="Cms.Extensions.GoogleGeoCoder.UserStrategy"/>
      </strategies>
    </add>
  </objectStrategies>
</objectFactory>
```

More information about ObjectFactory objectStrategies

- Register extensions in the `<objectStrategies>` element of the *siteroot*`/ObjectFactory.config` file.
- For the element directly below `<objectStrategies>`, `<addname>`, enter a `name` to identify extension type. The following table. Show the valid `name`s.

- List of Valid Names
  - Inventory
  - Coupon
  - BasketCalculator
  - CouponCalculator
  - ShippingCalculator
  - TaxCalculator
  - Country
  - Region
  - Basket
  - Blog
  - Order
  - Customer
  - CatalogEntry
  - CommunityGroup
  - Tag
  - Message Board
  - MicroMessage
  - Content
  - Folder
  - User
  - Forum
  - WebEvent
  - Taxonomy

- Within a `<strategies>` element, add one or more content extensions. For each extension, insert a `name` and `type`. For example:

```
<strategies>
 <add name="MyFirstExample"
   type="Cms.Extensions.Samples.ContentExtensionsSample"/>
</strategies>
```

  - `name` is free-text.
  - `Type` is a fully-qualified name of the extension you are registering. It consists of namespace + "." + name of the class.

Testing the strategy

The new extension (created in the examples) adds "modified" to the end of a new content item's title. To test this, create new content in the Workarea; "modified" will be appended to the end of the title.

Ektron.Cms.Extensibility strategies

An event is an activity that occurs within Ektron and is exposed through the Extension Framework. An *event handler* is a method that executes when an event occurs.

NOTE: `onBefore` lets you manipulate the current data *before* it is updated. In contrast, `onAfter` events do not let you change a value in the contentData object for the content being affected. `onAfter` events only update data somewhere else (such as within Ektron or a third-party database).

This section also contains the following topics.

ActivityCommentStrategy

Namespace: `Ektron.Cms.Extensibility`

- `ActivityCommentStrategy()`

  Summary: Constructor.

```
public ActivityCommentStrategy()
```

- `OnAfterDeleteActivityComment(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Activity Comment is deleted.

```
public virtual void
 OnAfterDeleteActivityComment(long id,
 Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateActivityComment (Ektron.Cms.Activity.ActivityCommentData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Activity Comment is edited.

```
public virtual void
  OnAfterUpdateActivityComment(Ektron.Cms.Activity.ActivityCommentData
  activity, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

ActivityStrategy

Namespace: `Ektron.Cms.Extensibility`

- `ActivityStrategy()`

  Summary: Constructor.

```
public ActivityStrategy()
```

- `OnAfterDeleteActivity(long, Ektron.Cms.Extensibility.CmsEventArgs)`

Summary: This method is called after a Activity is deleted

```
public virtual void OnAfterDeleteActivity(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateActivity(Ektron.Cms.Activity.ActivityData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Activity is edited.

```
public virtual void
  OnAfterUpdateActivity(Ektron.Cms.Activity.ActivityData activity,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

AdaptiveLibraryImages

Namespace: `Ektron.Cms.Extensibility`

- `Library`

  Summary: Gets a reference to the library manager.

```
public Ektron.Cms.Content.ILibraryManager Library { get; }
```

- `OnAfterAdd(Ektron.Cms.LibraryData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Generates the adaptive image cache for added library images.

```
public override void OnAfterAdd(Ektron.Cms.LibraryData eventItem,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.LibraryData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Updates the adaptive image cache for library images.

```
public override void OnAfterUpdate(Ektron.Cms.LibraryData eventItem,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Cleans up references to the library image from the adaptive image cache.

```
public override void OnBeforeDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

AssignPreviewDeviceBreakpointStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterAddDeviceBreakpoint (Ektron.Cms.Settings.Mobile.DeviceBreakpointData)`

  Summary: Assigns device previews from other breakpoints

```
public override void OnAfterAddDeviceBreakpoint(
  Ektron.Cms.Settings.Mobile.DeviceBreakpointData breakpoint)
```

- `OnAfterUpdateDeviceBreakpoint (Ektron.Cms.Settings.Mobile.DeviceBreakpointData)`

Summary: Confirms device previews currently assigned to the breakpoint and searches for previews from other breakpoints to assign.

```
public override void OnAfterUpdateDeviceBreakpoint(
  Ektron.Cms.Settings.Mobile.DeviceBreakpointData breakpoint)
```

- `OnBeforeDeleteDeviceBreakpoint(long)`

  Summary: Assigns device previews orphaned by the breakpoint being deleted to the next largest breakpoint or none.

```
public override void OnBeforeDeleteDeviceBreakpoint(long id)
```

BlogStrategy

Namespace: `Ektron.Cms.Extensibility.Content`

- `BlogStrategy()`

  Summary: Constructor.

```
public BlogStrategy()
```

- `OnAfterAddBlogComment(Ektron.Cms.BlogComment, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterAddBlogComment(Ektron.Cms.BlogComment comment,
  Ektron.Cms.Extensibility.CmsEventArgs args)
```

- `OnAfterAddBlogPost(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterAddBlogPost(Ektron.Cms.ContentData data,
  Ektron.Cms.Extensibility.CmsEventArgs args)
```

- `OnAfterPublishBlogPost(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterPublishBlogPost(Ektron.Cms.ContentData data,
  Ektron.Cms.Extensibility.CmsEventArgs args)
```

- `OnAfterUpdateBlogComment(Ektron.Cms.BlogComment, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterUpdateBlogComment(Ektron.Cms.BlogComment data,
  Ektron.Cms.Extensibility.CmsEventArgs args)
```

- `OnAfterUpdateBlogPost(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterUpdateBlogPost(Ektron.Cms.ContentData data,
  Ektron.Cms.Extensibility.CmsEventArgs args)Summary: Summary:
```

- `OnBeforeAddBlogPost(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeAddBlogPost(Ektron.Cms.ContentData data,
  Ektron.Cms.Extensibility.CmsEventArgs args)
```

- `OnBeforePublishBlogPost(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforePublishBlogPost(Ektron.Cms.ContentData data,
  Ektron.Cms.Extensibility.CmsEventArgs args)
```

- `OnBeforeUpdateBlogComment(Ektron.Cms.BlogComment, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeUpdateBlogComment(Ektron.Cms.BlogComment data,
  Ektron.Cms.Extensibility.CmsEventArgs args)
```

- `OnBeforeUpdateBlogPost(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeUpdateBlogPost(Ektron.Cms.ContentData data,
  Ektron.Cms.Extensibility.CmsEventArgs args)
```

CancellableEventArgs

Namespace: `Ektron.Cms.Extensibility`

- `CancellableEventArgs()`

  Summary: Constructor.

```
public CancellableEventArgs()
```

- `CancellableEventArgs(long)`

```
public CancellableEventArgs(long objectId)
```

- `CancellableEventArgs(long, EkRequestInformation)`

```
public CancellableEventArgs(long objectId,
  EkRequestInformation requestInformation)
```

- `CancellableEventArgs(long, EkRequestInformation, System.Data.Common.DbTransaction)`

```
public CancellableEventArgs(long objectId,
  EkRequestInformation requestInformation,
  System.Data.Common.DbTransaction transaction)
```

- `CancellationMessage`

  Summary: Gets or sets the message indicating the reason for the action cancellation.

```
public string CancellationMessage { set; get; }
```

- `IsCancelled`

  Summary: Gets or sets the is cancelled flag indicating whether or not the action raising the event should be cancelled.

```
public bool IsCancelled { set; get; }
```

CmsListStrategy

Namespace: `Ektron.Cms.Extensibility.Marketing`

- `CmsListStrategy()`

  Summary: Constructor.

```
public CmsListStrategy()
```

- `OnAfterAdd(Ektron.Cms.Marketing.ListData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnAfterAdd(Ektron.Cms.Marketing.ListData listdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnAfterDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterSubscriberAdded(long, Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnAfterSubscriberAdded(long listId,
  Ektron.Cms.Marketing.SubscriberData subscriberData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterSubscriberRemoved(long, Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnAfterSubscriberRemoved(long listId,
  Ektron.Cms.Marketing.SubscriberData subscriberData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.Marketing.ListData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnAfterUpdate(Ektron.Cms.Marketing.ListData listdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAdd(Ektron.Cms.Marketing.ListData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnBeforeAdd(Ektron.Cms.Marketing.ListData listdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnBeforeDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeSubscriberAdded(long, Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnBeforeSubscriberAdded(long listId,
  Ektron.Cms.Marketing.SubscriberData subscriberData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeSubscriberRemoved(long, Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnBeforeSubscriberRemoved(long listId,
  Ektron.Cms.Marketing.SubscriberData subscriberData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdate(Ektron.Cms.Marketing.ListData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnBeforeUpdate(Ektron.Cms.Marketing.ListData listdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

CmsMessageStrategy

Namespace: `Ektron.Cms.Extensibility`

- `CmsMessageStrategy()`

  Summary: Constructor.

```
public CmsMessageStrategy()
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a CmsMessageData object is deleted.

```
public virtual void OnAfterDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.Messaging.CmsMessageData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a CmsMessageData object is edited.

```
public virtual void
  OnAfterUpdate(Ektron.Cms.Messaging.CmsMessageData cmsMessageDataObject,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

CmsMessageTypeStrategy

Namespace: `Ektron.Cms.Extensibility`

- `CmsMessageTypeStrategy()`

  Summary: Constructor.

```
public CmsMessageTypeStrategy()
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a CmsMessageTypeData object is deleted.

```
public virtual void OnAfterDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.Messaging.CmsMessageTypeData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a CmsMessageTypeData object is edited.

```
public virtual void
  OnAfterUpdate(Ektron.Cms.Messaging.CmsMessageTypeData
  cmsMessageTypeDataObject, Ektron.Cms.Extensibility.CmsEventArgs
  eventArgs)
```

CmsSubscriberStrategy

Namespace: `Ektron.Cms.Extensibility.Marketing`

- `OnAfterAdd(Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnAfterAdd(Ektron.Cms.Marketing.SubscriberData
  subscriberdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void OnAfterDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnAfterUpdate(Ektron.Cms.Marketing.SubscriberData
  subscriberdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAdd(Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnBeforeAdd(Ektron.Cms.Marketing.SubscriberData
  subscriberdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnBeforeDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdate(Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public override void
  OnBeforeUpdate(Ektron.Cms.Marketing.SubscriberData
  subscriberdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

CollectionStrategy

Namespace: `Ektron.Cms.Extensibility`

- `CollectionStrategy()`

  Summary: Constructor.

```
public CollectionStrategy()
```

- `OnAfterDeleteCollection(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a collection is deleted.

```
public virtual void OnAfterDeleteCollection(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateCollection (Ektron.Cms.Organization.ContentCollectionData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a collection is edited

```
public virtual void
  OnAfterUpdateCollection(Ektron.Cms.Organization.ContentCollectionData
  collectionData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

CommunityGroupStrategy

Namespace: `Ektron.Cms.Extensibility`

- `CommunityGroupStrategy()`

  Summary: Constructor.

```
public CommunityGroupStrategy()
```

- `OnAdd(Ektron.Cms.CommunityGroupData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a new Community Group is added.

```
public virtual void OnAdd(Ektron.Cms.CommunityGroupData groupData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUserAdd(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user is added to a Community Group.

```
public virtual void OnAfterUserAdd(long CommunityGroupId,
  long userId, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUserDelete(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user is removed from a Community Group.

```
public virtual void OnAfterUserDelete(long CommunityGroupId,
  long userId, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Community Group is deleted.

```
public virtual void OnDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnUpdate(Ektron.Cms.CommunityGroupData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Community Group is updated.

```
public virtual void
  OnUpdate(Ektron.Cms.CommunityGroupData groupData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

ConfigurationStrategy

Namespace: `Ektron.Cms.Extensibility`

- `ConfigurationStrategy()`

  Summary: Constructor.

  ```
  public ConfigurationStrategy()
  ```

- `OnAfterAddConfiguration(System.Collections.Hashtable, Ektron.Cms.Extensibility.CmsEventArgs)`

  ```
  public virtual void
    OnAfterAddConfiguration(System.Collections.Hashtable siteVars,
    Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
  ```

- `OnAfterDeleteConfiguration(System.Collections.Hashtable, Ektron.Cms.Extensibility.CmsEventArgs)`

  ```
  public virtual void
    OnAfterDeleteConfiguration(System.Collections.Hashtable siteVars,
    Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
  ```

- `OnAfterUpdateConfiguration(System.Collections.Hashtable, Ektron.Cms.Extensibility.CmsEventArgs)`

  ```
  public virtual void
    OnAfterUpdateConfiguration(System.Collections.Hashtable siteVars,
    Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
  ```

- `OnBeforeAddConfiguration(System.Collections.Hashtable, Ektron.Cms.Extensibility.CmsEventArgs)`

  ```
  public virtual void
    OnBeforeAddConfiguration(System.Collections.Hashtable siteVars,
    Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
  ```

- `OnBeforeDeleteConfiguration(System.Collections.Hashtable, Ektron.Cms.Extensibility.CmsEventArgs)`

  ```
  public virtual void
    OnBeforeDeleteConfiguration(System.Collections.Hashtable siteVars,
    Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
  ```

- `OnBeforeUpdateConfiguration(System.Collections.Hashtable, Ektron.Cms.Extensibility.CmsEventArgs)`

  ```
  public virtual void
    OnBeforeUpdateConfiguration(System.Collections.Hashtable siteVars,
    Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
  ```

ContentRatingStrategy

Namespace: `Ektron.Cms.Extensibility`

- `ContentRatingStrategy()`

  Summary: Constructor.

```
public ContentRatingStrategy()
```

- `OnAfterDeleteContentRating(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Content Rating is deleted.

```
public virtual void OnAfterDeleteContentRating(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateContentRating(Ektron.Cms.ContentRatingData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Content Rating is edited.

```
public virtual void
  OnAfterUpdateContentRating(Ektron.Cms.ContentRatingData
  contentRating, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

ContentStrategy

Namespace: `Ektron.Cms.Extensibility`

- `ContentStrategy()`

  Summary: Constructor.

```
public ContentStrategy()
```

- `OnAfterAddContent(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after new content is added.

```
public virtual void
  OnAfterAddContent(Ektron.Cms.ContentData contentData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterContentArchive(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual
  void OnAfterContentArchive(Ektron.Cms.ContentData contentData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterCopyContent(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterCopyContent(long contentId, long newId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterCopyContent(long, long, int, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterCopyContent(long contentId, long newId,
  int languageid, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDeleteContent(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after content is deleted.

```
public virtual void OnAfterDeleteContent(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterMoveContent(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterMoveContent(long contentId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterPublishContent(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after content is published.

```
public virtual void
  OnAfterPublishContent(Ektron.Cms.ContentData contentData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterSubmitContent(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterSubmitContent(long contentId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateContent(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after content is saved.

```
public virtual void
  OnAfterUpdateContent(Ektron.Cms.ContentData contentData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddContent(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before content is added.

```
public virtual void OnBeforeAddContent(Ektron.Cms.ContentData
  contentData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeContentArchive(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeContentArchive(Ektron.Cms.ContentData
  contentData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeCopyContent(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeCopyContent(long contentId, long newId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteContent(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before content is deleted.

```
public virtual void OnBeforeDeleteContent(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeMoveContent(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeMoveContent(long contentId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforePublishContent(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

Summary: This method is called before content is published.

```
public virtual void
  OnBeforePublishContent(Ektron.Cms.ContentData contentData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeSubmitContent(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeSubmitContent(long contentId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateContent(Ektron.Cms.ContentData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before content is saved.

```
public virtual void
  OnBeforeUpdateContent(Ektron.Cms.ContentData contentData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

CustomFieldStrategy

Namespace: `Ektron.Cms.Extensibility.Marketing`

- `CustomFieldStrategy()`

  Summary: Constructor.

```
public CustomFieldStrategy()
```

- `OnAfterAdd(Ektron.Cms.Marketing.CustomFieldData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterAdd(Ektron.Cms.Marketing.CustomFieldData
  customfielddata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.Marketing.CustomFieldData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterUpdate(Ektron.Cms.Marketing.CustomFieldData
  customfielddata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAdd(Ektron.Cms.Marketing.CustomFieldData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeAdd(Ektron.Cms.Marketing.CustomFieldData
  customfielddata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnBeforeUpdate(Ektron.Cms.Marketing.CustomFieldData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void
  OnBeforeUpdate(Ektron.Cms.Marketing.CustomFieldData
  customfielddata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

CustomPropertyStrategy

Namespace: Ektron.Cms.Extensibility

- OnAfterAddCustomProperty(Ektron.Cms.Common.CustomPropertyData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void
  OnAfterAddCustomProperty(Ektron.Cms.Common.CustomPropertyData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterAddCustomPropertyObject (Ektron.Cms.Common.CustomPropertyObjectData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void
  OnAfterAddCustomPropertyObject(Ektron.Cms.Common.CustomPropertyObjectData
  data, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterDeleteCustomProperty(long, int, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void OnAfterDeleteCustomProperty(long propertyId,
  int languageId, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterDeleteCustomPropertyObject(long, int, EkEnumeration.CustomPropertyObjectType, long, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void OnAfterDeleteCustomPropertyObject(long objectId,
  int languageId, EkEnumeration.CustomPropertyObjectType cmsObjectType,
  long propertyId, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterUpdateCustomProperty(Ektron.Cms.Common.CustomPropertyData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void
  OnAfterUpdateCustomProperty(Ektron.Cms.Common.CustomPropertyData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterUpdateCustomPropertyObject (Ektron.Cms.Common.CustomPropertyObjectData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void
  OnAfterUpdateCustomPropertyObject(Ektron.Cms.Common.CustomPropertyObjectData
  data, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnBeforeAddCustomProperty(Ektron.Cms.Common.CustomPropertyData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void
  OnBeforeAddCustomProperty(Ektron.Cms.Common.CustomPropertyData
  data, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddCustomPropertyObject (Ektron.Cms.Common.CustomPropertyObjectData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeAddCustomPropertyObject(Ektron.Cms.Common.CustomPropertyObjectData
  data, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteCustomProperty(long, int, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeDeleteCustomProperty(long propertyId, int languageId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteCustomPropertyObject(long, int, EkEnumeration.CustomPropertyObjectType, long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeDeleteCustomPropertyObject(long objectId, int languageId,
  EkEnumeration.CustomPropertyObjectType cmsObjectType, long propertyId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateCustomProperty (Ektron.Cms.Common.CustomPropertyData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeUpdateCustomProperty(Ektron.Cms.Common.CustomPropertyData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateCustomPropertyObject (Ektron.Cms.Common.CustomPropertyObjectData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeUpdateCustomPropertyObject(Ektron.Cms.Common.CustomPropertyObjectData
  data, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

DeviceBreakpointStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterAddDeviceBreakpoint (Ektron.Cms.Settings.Mobile.DeviceBreakpointData)`

  Summary: This method is called after a device breakpoint is added.

```
public virtual void
  OnAfterAddDeviceBreakpoint(Ektron.Cms.Settings.Mobile.DeviceBreakpointData
  breakpoint)
```

- `OnAfterDeleteDeviceBreakpoint(long)`

  Summary: This method is called after a device breakpoint is deleted.

```
public virtual void OnAfterDeleteDeviceBreakpoint(long id)
```

- `OnAfterUpdateDeviceBreakpoint (Ektron.Cms.Settings.Mobile.DeviceBreakpointData)`

Summary: This method is called after a device breakpoint is updated.

```
public virtual void
  OnAfterUpdateDeviceBreakpoint(Ektron.Cms.Settings.Mobile.DeviceBreakpointData
  breakpoint)
```

- `OnBeforeAddDeviceBreakpoint (Ektron.Cms.Settings.Mobile.DeviceBreakpointData)`

  Summary: This method is called before a device breakpoint is added.

```
public virtual void
  OnBeforeAddDeviceBreakpoint(Ektron.Cms.Settings.Mobile.DeviceBreakpointData
  breakpoint)
```

- `OnBeforeDeleteDeviceBreakpoint(long)`

  Summary: This method is called before a device breakpoint is deleted.

```
public virtual void OnBeforeDeleteDeviceBreakpoint(long id)
```

- `OnBeforeUpdateDeviceBreakpoint (Ektron.Cms.Settings.Mobile.DeviceBreakpointData)`

  Summary: This method is called before a device breakpoint is updated.

```
public virtual void
  OnBeforeUpdateDeviceBreakpoint(Ektron.Cms.Settings.Mobile.DeviceBreakpointData
  breakpoint)
```

ESyncDataTransformStrategy

Namespace: `Ektron.Cms.Extensibility`

9.00 and higher

- `OnAfterDataRowSelected(System.Data.DataRow, Ektron.Cms.CoreService.SyncDataTransformContext)`

  Summary: This method is called after the sync framework has determined the rows to be synced as per the sync criteria. By design, this event is raised synchronously for every row in the DataTable, so that there is minimal impact on eSync memory consumption or performance. *disconnectedRow:* A row corresponding to the actual row which will be synced. Any changes to this will be propagated back to the original row. Note however that this is disconnected from the actual DataRow object and does not provide access to the real DataTable or Dataset being used by the sync framework. Only row value changes are supported for transformation.

  context: This parameter provides more context about the DataRow being passed.

```
public virtual void
  OnAfterDataRowSelected(System.Data.DataRow disconnectedRow,
  Ektron.Cms.CoreService.SyncDataTransformContext context)
```

ESyncNotificationStrategy

Namespace: `Ektron.Cms.Extensibility`

9.00 and higher

- `OnAfterDatabaseSyncRunCompleteFromLocal (Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after a Database Sync initiated by an eSync run has completed. This method is provided because some eSync types can cause multiple database syncs (Ex: Package Sync, UGC Sync). This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework. Not all properties of the SyncProfile parameter are populated for this event. This method is called even if the database sync errors out.

  ```
  public virtual void
    OnAfterDatabaseSyncRunCompleteFromLocal(Ektron.Cms.CoreService.SyncProfile
    syncProfileData)
  ```

- `OnAfterDatabaseSyncRunCompleteFromPeer (Ektron.Cms.CoreService.PeerServerDetails, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after a Database Sync initiated by a peer eSync run has completed. This happens after OnAfterDatabaseSyncRunCompleteFromLocal() has been raised on the peer server. This method is provided because an eSync can cause multiple database syncs (Ex: Package Sync, UGC Sync). This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework. Not all properties of the SyncProfile parameter are populated for this event. This method is called even if the database sync errors out.

  ```
  public virtual void
    OnAfterDatabaseSyncRunCompleteFromPeer(Ektron.Cms.CoreService.PeerServerDetails
    peerDetails, Ektron.Cms.CoreService.SyncProfile syncProfileData)
  ```

- `OnAfterDatabaseSyncRunErrorFromLocal(System.Exception, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after a Database Sync initiated by an eSync run has errored out. This method is provided because some eSync types can cause multiple database syncs (Ex: Package Sync, UGC Sync). This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework. Not all properties of the SyncProfile parameter are populated for this event.

  ```
  public virtual void
    OnAfterDatabaseSyncRunErrorFromLocal
    (System.Exception ex, Ektron.Cms.CoreService.SyncProfile syncProfileData)
  ```

- `OnAfterDatabaseSyncRunErrorFromPeer(System.Exception, Ektron.Cms.CoreService.PeerServerDetails, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after a Database Sync initiated by a peer eSync run has errored out. This happens after OnAfterDatabaseSyncRunCompleteFromLocal() has been raised on the peer server. This method is provided because an eSync can cause multiple database

syncs (Ex: Package Sync, UGC Sync). This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework. Not all properties of the SyncProfile parameter are populated for this event.

```
public virtual void
  OnAfterDatabaseSyncRunErrorFromPeer(System.Exception ex,
  Ektron.Cms.CoreService.PeerServerDetails peerDetails,
  Ektron.Cms.CoreService.SyncProfile syncProfileData)
```

- `OnAfterDatabaseSyncRunStartFromLocal (Ektron.Cms.CoreService.SyncProfile, Ektron.Cms.Extensibility.SyncCommandEventArgs)`

  Summary: This method is called after a Database Sync initiated by an eSync run has started. This method is provided because an eSync can cause multiple database syncs (Ex: Package Sync, UGC Sync). This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework. Not all properties of the SyncProfile parameter are populated for this event. This event is cancellable and properties can be propagated back to EWS via the syncProfileData parameter.

  ```
  public virtual void
    OnAfterDatabaseSyncRunStartFromLocal(Ektron.Cms.CoreService.SyncProfile
    syncProfileData, Ektron.Cms.Extensibility.SyncCommandEventArgs commandArgs)
  ```

- `OnAfterDatabaseSyncRunStartFromPeer (Ektron.Cms.CoreService.PeerServerDetails, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after a Database Sync initiated by a peer eSync run has started. This happens after OnAfterDatabaseSyncRunStartFromLocal() has been raised on the peer server. This method is provided because an eSync can cause multiple database syncs (Ex: Package Sync, UGC Sync). This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework. Not all properties of the SyncProfile parameter are populated for this event.

  ```
  public virtual void
    OnAfterDatabaseSyncRunStartFromPeer(Ektron.Cms.CoreService.PeerServerDetails
    peerDetails, Ektron.Cms.CoreService.SyncProfile syncProfileData)
  ```

- `OnAfterESyncRunCompleteFromLocal (Ektron.Cms.CoreService.SyncSpecificDetails, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after an eSync run orchestrated by the local Ektron Windows Service has completed. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework. This method is called even if eSync errors out.

  ```
  public virtual void
    OnAfterESyncRunCompleteFromLocal(Ektron.Cms.CoreService.SyncSpecificDetails
    details, Ektron.Cms.CoreService.SyncProfile syncProfileData)
  ```

- `OnAfterESyncRunCompleteFromPeer (Ektron.Cms.CoreService.PeerServerDetails, Ektron.Cms.CoreService.SyncSpecificDetails,`

Ektron.Cms.CoreService.SyncProfile)

Summary: This method is called after an eSync run orchestrated by a peer Ektron Windows Service has completed. This happens after OnAfterESyncRunCompleteFromLocal() has been raised on the peer server. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework. This method is called even if eSync errors out.

```
public virtual void
  OnAfterESyncRunCompleteFromPeer(Ektron.Cms.CoreService.PeerServerDetails
  peerDetails, Ektron.Cms.CoreService.SyncSpecificDetails details,
  Ektron.Cms.CoreService.SyncProfile syncProfileData)
```

- OnAfterESyncRunErrorFromLocal(System.Exception, Ektron.Cms.CoreService.SyncSpecificDetails, Ektron.Cms.CoreService.SyncProfile)

  Summary: This method is called after an eSync run orchestrated by the local Ektron Windows Service has errored out. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework.

  ```
  public virtual void
    OnAfterESyncRunErrorFromLocal(System.Exception ex,
    Ektron.Cms.CoreService.SyncSpecificDetails details,
    Ektron.Cms.CoreService.SyncProfile syncProfileData)
  ```

- OnAfterESyncRunErrorFromPeer(System.Exception, Ektron.Cms.CoreService.PeerServerDetails, Ektron.Cms.CoreService.SyncSpecificDetails, Ektron.Cms.CoreService.SyncProfile)

  Summary: This method is called after an eSync run orchestrated by a peer Ektron Windows Service has errored out.This happens after OnAfterESyncRunErrorFromLocal() has been raised on the peer server. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework.

  ```
  public virtual void
    OnAfterESyncRunErrorFromPeer(System.Exception ex,
    Ektron.Cms.CoreService.PeerServerDetails peerDetails,
    Ektron.Cms.CoreService.SyncSpecificDetails details,
    Ektron.Cms.CoreService.SyncProfile syncProfileData)
  ```

- OnAfterESyncRunStartFromLocal (Ektron.Cms.CoreService.SyncSpecificDetails, Ektron.Cms.CoreService.SyncProfile, Ektron.Cms.Extensibility.SyncCommandEventArgs)

  Summary: This method is called after an eSync run orchestrated by the local Ektron Windows Service has started. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework.

  ```
  public virtual void
    OnAfterESyncRunStartFromLocal(Ektron.Cms.CoreService.SyncSpecificDetails
    details, Ektron.Cms.CoreService.SyncProfile syncProfileData,
    Ektron.Cms.Extensibility.SyncCommandEventArgs commandArgs)
  ```

- `OnAfterESyncRunStartFromPeer (Ektron.Cms.CoreService.PeerServerDetails, Ektron.Cms.CoreService.SyncSpecificDetails, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after an eSync run orchestrated by a peer Ektron Windows Service has started. This happens after `OnAfterESyncRunStartFromLocal()` has been raised on the peer server. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework.

```
public virtual void
  OnAfterESyncRunStartFromPeer(Ektron.Cms.CoreService.PeerServerDetails
  peerDetails, Ektron.Cms.CoreService.SyncSpecificDetails details,
  Ektron.Cms.CoreService.SyncProfile syncProfileData)
```

- `OnAfterFolderSyncRunCompleteFromLocal (Ektron.Cms.CoreService.FolderSyncProfile, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after a Folder Sync initiated by an eSync run has completed. This method is provided because some eSync types can cause multiple folder syncs. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework. This method is called even if the database sync errors out.

```
public virtual void
  OnAfterFolderSyncRunCompleteFromLocal
  (Ektron.Cms.CoreService.FolderSyncProfile folderSyncProfileData,
  Ektron.Cms.CoreService.SyncProfile syncProfileData)
```

- `OnAfterFolderSyncRunCompleteFromPeer (Ektron.Cms.CoreService.PeerServerDetails, Ektron.Cms.CoreService.FolderSyncProfile, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after a Folder Sync initiated by a peer eSync run has completed. This happens after `OnAfterFolderSyncRunCompleteFromPeer()` has been raised on the peer server. This method is provided because some eSync types can cause multiple folder syncs. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework. This method is called even if the file sync errors out.

```
public virtual void
  OnAfterFolderSyncRunCompleteFromPeer(Ektron.Cms.CoreService.PeerServerDetails
  peerDetails, Ektron.Cms.CoreService.FolderSyncProfile
  folderSyncProfileData, Ektron.Cms.CoreService.SyncProfile syncProfileData)
```

- `OnAfterFolderSyncRunErrorFromLocal(System.Exception, Ektron.Cms.CoreService.FolderSyncProfile, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after a Folder Sync initiated by an eSync run has errored out. This method is provided because some eSync types can cause

multiple folder syncs. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework.

```
public virtual void
  OnAfterFolderSyncRunErrorFromLocal(System.Exception ex,
  Ektron.Cms.CoreService.FolderSyncProfile folderSyncProfileData,
  Ektron.Cms.CoreService.SyncProfile syncProfileData)
```

- `OnAfterFolderSyncRunErrorFromPeer(System.Exception, Ektron.Cms.CoreService.PeerServerDetails, Ektron.Cms.CoreService.FolderSyncProfile, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after a Folder Sync initiated by a peer eSync run has errored out. This happens after `OnAfterFolderSyncRunErrorFromLocal()` has been raised on the peer server. This method is provided because some eSync types can cause multiple folder syncs. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework.

  ```
  public virtual void
    OnAfterFolderSyncRunErrorFromPeer(System.Exception ex,
    Ektron.Cms.CoreService.PeerServerDetails peerDetails,
    Ektron.Cms.CoreService.FolderSyncProfile folderSyncProfileData,
    Ektron.Cms.CoreService.SyncProfile syncProfileData)
  ```

- `OnAfterFolderSyncRunStartFromLocal (Ektron.Cms.CoreService.FolderSyncProfile, Ektron.Cms.CoreService.SyncProfile, Ektron.Cms.Extensibility.SyncCommandEventArgs)`

  Summary: This method is called after a Folder Sync initiated by an eSync run has started. This method is provided because some eSync types can cause multiple folder syncs. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework.

  ```
  public virtual void
    OnAfterFolderSyncRunStartFromLocal(Ektron.Cms.CoreService.FolderSyncProfile
    folderSyncProfileData, Ektron.Cms.CoreService.SyncProfile syncProfileData,
    Ektron.Cms.Extensibility.SyncCommandEventArgs commandArgs)
  ```

- `OnAfterFolderSyncRunStartFromPeer (Ektron.Cms.CoreService.PeerServerDetails, Ektron.Cms.CoreService.FolderSyncProfile, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called after a Folder Sync initiated by a peer eSync run has started. This happens after `OnAfterFolderSyncRunStartFromLocal()` has been raised on the peer server. This method is provided because some eSync types can cause multiple folder syncs. This method is NOT initiated by the actual synchronization process being performed by the MS Sync Framework.

  ```
  public virtual void
    OnAfterFolderSyncRunStartFromPeer(Ektron.Cms.CoreService.PeerServerDetails
    peerDetails, Ektron.Cms.CoreService.FolderSyncProfile folderSyncProfileData,
    Ektron.Cms.CoreService.SyncProfile syncProfileData)
  ```

- `OnLocalDatabaseSyncProgress (Ektron.Cms.CoreService.DatabaseSyncProgress, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called when a database sync initiated locally is running. This method is initiated by the actual synchronization process being performed by the MS Sync Framework. Not all properties of the SyncProfile parameter are populated in a progress event.

```
public virtual void
  OnLocalDatabaseSyncProgress(Ektron.Cms.CoreService.DatabaseSyncProgress
  syncProgress, Ektron.Cms.CoreService.SyncProfile syncProfileData)
```

- `OnLocalFileSyncProgress(Ektron.Cms.CoreService.FileSyncProgress, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called when a file sync initiated locally is running. This event is not raised for all file syncs. This method is initiated by the actual synchronization process being performed by the MS Sync Framework. The behavior of this event corresponds directly to the behavior of the MS Sync Framework 'SessionProgress' and 'StateChanged' events of the SyncOrchestrator class. Not all properties of the SyncProfile parameter are populated in a progress event.

```
public virtual void
  OnLocalFileSyncProgress(Ektron.Cms.CoreService.FileSyncProgress
  syncProgress, Ektron.Cms.CoreService.SyncProfile syncProfileData)
```

- `OnPeerDatabaseSyncProgress (Ektron.Cms.CoreService.DatabaseSyncProgress, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called when a database sync initiated by a peer is running. This happens when the peer server is running database sync to the local server as part of the eSync process. This method is initiated by the actual synchronization process being performed by the MS Sync Framework. Not all properties of the SyncProfile parameter are populated in a progress event.

```
public virtual void
  OnPeerDatabaseSyncProgress(Ektron.Cms.CoreService.DatabaseSyncProgress
  syncProgress, Ektron.Cms.CoreService.SyncProfile syncProfileData)
```

- `OnPeerFileSyncProgress(Ektron.Cms.CoreService.FileSyncProgress, Ektron.Cms.CoreService.SyncProfile)`

  Summary: This method is called when a file sync initiated by a peer is running. This happens when the peer server is running file sync to the local server as part of the eSync or Load Balancing process. This method is initiated by the actual synchronization process being performed by the MS Sync Framework. Not all properties of the SyncProfile parameter are populated in a progress event.

```
public virtual void
  OnPeerFileSyncProgress(Ektron.Cms.CoreService.FileSyncProgress
  syncProgress, Ektron.Cms.CoreService.SyncProfile syncProfileData)
```

FavoriteStrategy

Namespace: `Ektron.Cms.Extensibility`

- `FavoriteStrategy()`

  Summary: Constructor.

```
public FavoriteStrategy()
```

- `OnAfterDeleteFavorite(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a favorite is deleted.

```
public virtual void
  OnAfterDeleteFavorite(long id, Ektron.Cms.Extensibility.CmsEventArgs
  eventArgs)
```

- `OnAfterUpdateFavorite(Ektron.Cms.Community.FavoriteItemData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a favorite is edited.

```
public virtual void
  OnAfterUpdateFavorite(Ektron.Cms.Community.FavoriteItemData
  favoriteItemData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

FavoriteTaxonomyStrategy

Namespace: `Ektron.Cms.Extensibility`

- `FavoriteTaxonomyStrategy()`

  Summary: Constructor.

```
public abstract class FavoriteTaxonomyStrategy
```

- `OnAfterDeleteFavoriteTaxonomy(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Favorite Taxonomy is deleted.

```
public virtual void OnAfterDeleteFavoriteTaxonomy(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateFavoriteTaxonomy (Ektron.Cms.Community.FavoriteTaxonomyData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Favorite Taxonomy is edited.

```
public virtual void
  OnAfterUpdateFavoriteTaxonomy(Ektron.Cms.Community.FavoriteTaxonomyData
  favoriteTaxonomyData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

FlagStrategy

Namespace: `Ektron.Cms.Extensibility`

- `FlagStrategy()`

  Summary: Constructor.

```
public FlagStrategy()
```

- `OnAfterAddFlag(Ektron.Cms.ObjectFlagData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterAddFlag(Ektron.Cms.ObjectFlagData flagData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDeleteFlag(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterDeleteFlag(long Id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateFlag(Ektron.Cms.ObjectFlagData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterUpdateFlag(Ektron.Cms.ObjectFlagData flagData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddFlag(Ektron.Cms.ObjectFlagData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeAddFlag(Ektron.Cms.ObjectFlagData flagData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteFlag(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeDeleteFlag(long Id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateFlag(Ektron.Cms.ObjectFlagData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeUpdateFlag(Ektron.Cms.ObjectFlagData flagData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

FolderStrategy

Namespace: `Ektron.Cms.Extensibility`

- `FolderStrategy()`

  Summary: Constructor.

```
public FolderStrategy()
```

- `OnAfterAddFolder(Ektron.Cms.FolderData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a new folder is added.

```
public virtual void
  OnAfterAddFolder(Ektron.Cms.FolderData folderData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterCopy(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterCopy(long oldFolderID, long newFolderID,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDeleteFolder(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a folder is deleted.

```
public virtual void
  OnAfterDeleteFolder(long Id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterMove(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterMove(long folderID, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateFolder(Ektron.Cms.FolderData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after folder is updated.

```
public virtual void
  OnAfterUpdateFolder(Ektron.Cms.FolderData folderData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddFolder(Ektron.Cms.FolderData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a folder is added.

```
public virtual void
  OnBeforeAddFolder(Ektron.Cms.FolderData folderData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeCopy(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeCopy(long oldFolderID, long newFolderID,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteFolder(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a folder is deleted.

```
public virtual void OnBeforeDeleteFolder(long Id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeMove(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeMove(long folderID,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateFolder(Ektron.Cms.FolderData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before changes to a folder are saved.

```
public virtual void OnBeforeUpdateFolder(Ektron.Cms.FolderData folderData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

FormStrategy

Namespace: `Ektron.Cms.Extensibility.Content`

- `FormStrategy()`

  Summary: Constructor.

```
public ForumStrategy()
```

- `OnAfterSubmit(Ektron.Cms.FormData, Ektron.Cms.FormSubmittedData, string, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterSubmit(Ektron.Cms.FormData formData,
  Ektron.Cms.FormSubmittedData submittedFormData, string formXml,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeSubmit(Ektron.Cms.FormData, Ektron.Cms.FormSubmittedData, string, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeSubmit(Ektron.Cms.FormData formData,
  Ektron.Cms.FormSubmittedData submittedFormData, string formXml,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

ForumStrategy

Namespace: `Ektron.Cms.Extensibility`

- `ForumStrategy()`

  Summary: Constructor.

```
public ForumStrategy()
```

- `OnAfterAddTopic(Ektron.Cms.DiscussionTopic, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterAddTopic(Ektron.Cms.DiscussionTopic topic,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterAddTopicReply(Ektron.Cms.TaskData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterAddTopicReply(Ektron.Cms.TaskData topicReply,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateTopic(Ektron.Cms.DiscussionTopic, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterUpdateTopic(Ektron.Cms.DiscussionTopic topic,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddTopic(Ektron.Cms.DiscussionTopic, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeAddTopic(Ektron.Cms.DiscussionTopic topic,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddTopicReply(Ektron.Cms.TaskData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeAddTopicReply(Ektron.Cms.TaskData topicReply,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateTopic(Ektron.Cms.DiscussionTopic, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeUpdateTopic(Ektron.Cms.DiscussionTopic topic,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

FriendsStrategy

Namespace: `Ektron.Cms.Extensibility`

- `FriendsStrategy()`

  Summary: Constructor.

```
public FriendsStrategy()
```

- `OnAfterDeleteFriend(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a friend is deleted.

```
public virtual void OnAfterDeleteFriend(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateFriend(Ektron.Cms.Community.FriendsData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a friend is edited.

```
public virtual void
  OnAfterUpdateFriend(Ektron.Cms.Community.FriendsData friendsData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

FriendsTaxonomyStrategy

Namespace: `Ektron.Cms.Extensibility`

- `FriendsTaxonomyStrategy()`

  Summary: Constructor.

```
public FriendsTaxonomyStrategy()
```

- `OnAfterDeleteFriendTaxonomy(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a friend taxonomy is deleted.

```
public virtual void OnAfterDeleteFriendTaxonomy(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateFriendTaxonomy (Ektron.Cms.Community.FriendTaxonomyData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a friend taxonomy is edited.

```
public virtual void
  OnAfterUpdateFriendTaxonomy(Ektron.Cms.Community.FriendTaxonomyData
  friendTaxonomyData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

GenericPreviewDeviceBreakpointStrategy

Namespace: `Ektron.Cms.Extensibility`

- `GenericPreviewDeviceBreakpointStrategy()`

  Summary: Constructor. Initializes a new instance of the GenericPreviewDeviceBreakpointStrategy class and populates its internal data.

```
public GenericPreviewDeviceBreakpointStrategy()
```

- `OnAfterAddDeviceBreakpoint (Ektron.Cms.Settings.Mobile.DeviceBreakpointData)`

  Summary: Create a new device preview for the breakpoint for each available device preview template.

```
public override void OnAfterAddDeviceBreakpoint(
  Ektron.Cms.Settings.Mobile.DeviceBreakpointData breakpoint)
```

- `OnAfterUpdateDeviceBreakpoint (Ektron.Cms.Settings.Mobile.DeviceBreakpointData)`

  Summary: Updates preview devices for the breakpoint which have been created from a device preview template so their width matches the breakpoint.

```
public override void OnAfterUpdateDeviceBreakpoint(
  Ektron.Cms.Settings.Mobile.DeviceBreakpointData breakpoint)
```

- `OnBeforeDeleteDeviceBreakpoint(long)`

  Summary: Deletes preview devices for the breakpoint which have been created from a device preview template.

```
public override void OnBeforeDeleteDeviceBreakpoint(long id)
```

LibraryStrategy

Namespace: `Ektron.Cms.Extensibility`

- `LibraryStrategy()`

  Summary: Constructor.

```
public LibraryStrategy()
```

- `OnAfterAdd(Ektron.Cms.LibraryData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called after a library item is added.

```
public virtual void
  OnAfterAdd(Ektron.Cms.LibraryData taxonomyData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called after a library item is deleted.

```
public virtual void
  OnAfterDelete(long id, Ektron.Cms.Extensibility.CmsEventArgs
  eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.LibraryData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called after a library item is updated.

```
public virtual void
  OnAfterUpdate(Ektron.Cms.LibraryData taxonomyData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAdd(Ektron.Cms.LibraryData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called before a library item is added.

```
public virtual void
  OnBeforeAdd(Ektron.Cms.LibraryData taxonomyData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called before a library item is deleted.

```
public virtual void
  OnBeforeDelete(long id, Ektron.Cms.Extensibility.CmsEventArgs
  eventArgs)
```

- `OnBeforeUpdate(Ektron.Cms.LibraryData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called before a library item is updated.

```
public virtual void
  OnBeforeUpdate(Ektron.Cms.LibraryData taxonomyData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

ListStrategy

Namespace: `Ektron.Cms.Extensibility.Marketing`

- `ListStrategy()`

  Summary: Constructor.

```
public ListStrategy()
```

- `OnAfterAdd(Ektron.Cms.Marketing.ListData, Ektron.Cms.Extensibility.CmsEventArgs))`

  Summary: Called after a list item is added.

```
public virtual void
  OnAfterAdd(Ektron.Cms.Marketing.ListData listdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called after a list item is deleted.

```
public virtual void
  OnAfterDelete(long id, Ektron.Cms.Extensibility.CmsEventArgs
  eventArgs)
```

- `OnAfterSubscriberAdded(long, Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterSubscriberAdded(long listId,
  Ektron.Cms.Marketing.SubscriberData subscriberData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterSubscriberRemoved(long, Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterSubscriberRemoved(long listId,
  Ektron.Cms.Marketing.SubscriberData subscriberData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.ListData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called after a list item is updated.

```
public virtual void
  OnAfterUpdate(Ektron.Cms.ListData listData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAdd(Ektron.Cms.ListData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called before a list item is added.

```
public virtual void
  OnBeforeAdd(Ektron.Cms.ListData listdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called before a list item is deleted.

```
public virtual void
  OnBeforeDelete(long id, Ektron.Cms.Extensibility.CmsEventArgs
  eventArgs)
```

- `OnBeforeSubscriberAdded(long, Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeSubscriberAdded(long listId,
  Ektron.Cms.Marketing.SubscriberData subscriberData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeSubscriberRemoved(long, Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeSubscriberRemoved(long listId,
  Ektron.Cms.Marketing.SubscriberData subscriberData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdate(Ektron.Cms.ListData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called before a list item is updated.

```
public virtual void
  OnBeforeUpdate(Ektron.Cms.ListData listdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

LoadBalancerNotificationStrategy

Namespace: `Ektron.Cms.Extensibility`

- `LoadBalancerNotificationStrategy()`

  Summary: Constructor.

```
protected LoadBalancerNotificationStrategy()
```

- `OnAfterRunComplete (System.Collections.Generic.List<Ektron.Cms.CoreService.LoadBalancedFile>, Ektron.Cms.CoreService.LoadBalanceProfile)`

  Summary: This method is called after the local load balancer synchronization session has completed successfully. If the session errored out, this event will not be raised.

```
public virtual void
  OnAfterRunComplete(System.Collections.Generic.List<LoadBalancedFile>
  files, Ektron.Cms.CoreService.LoadBalanceProfile profile)
```

- `OnAfterRunError(System.Exception, System.Collections.Generic.List<Ektron.Cms.CoreService.LoadBalancedFile>, Ektron.Cms.CoreService.LoadBalanceProfile)`

  Summary: This method is called after the local load balancer synchronization session has errored out.

```
public virtual void OnAfterRunError(System.Exception ex,
  System.Collections.Generic.List<LoadBalancedFile> files,
  Ektron.Cms.CoreService.LoadBalanceProfile profile)
```

- `OnBeforeRunStart (System.Collections.Generic.List<Ektron.Cms.CoreService.LoadBalancedFile>, Ektron.Cms.CoreService.LoadBalanceProfile)`

  Summary: This method is called before the local load balancer starts syncing the files in the queue.

```
public virtual void
  OnBeforeRunStart(System.Collections.Generic.List<LoadBalancedFile>
  files, Ektron.Cms.CoreService.LoadBalanceProfile profile)
```

LocaleStrategy

Namespace: `Ektron.Cms.Extensibility`

- `DetermineRequestedLocale(System.Web.HttpRequest, System.Web.HttpCookie, ref Ektron.Cms.Localization.LocaleData, Ektron.Cms.BusinessObjects.Localization.ILocaleManager, Ektron.Cms.Extensibility.CancellableEventArgs)`

  Summary: Determines the locale given the HttpRequest and HttpCookie.

```
public virtual void
  DetermineRequestedLocale(System.Web.HttpRequest request,
  System.Web.HttpCookie cookie,
  ref Ektron.Cms.Localization.LocaleData requestedLocale,
```

```
  Ektron.Cms.BusinessObjects.Localization.ILocaleManager sender,
  Ektron.Cms.Extensibility.CancellableEventArgs eventArgs)
```

- `LocaleStrategy()`

  Summary: Constructor. Initializes a new instance of the LocaleStrategy class.

```
public LocaleStrategy()
```

- `OnAfterAddLocale(Ektron.Cms.Localization.LocaleData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterAddLocale(Ektron.Cms.Localization.LocaleData localeData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDeleteLocale(Ektron.Cms.Localization.LocaleData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterDeleteLocale(Ektron.Cms.Localization.LocaleData localeData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDisableLocale(Ektron.Cms.Localization.LocaleData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterDisableLocale(Ektron.Cms.Localization.LocaleData localeData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterEditLocale(Ektron.Cms.Localization.LocaleData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnAfterEditLocale(Ektron.Cms.Localization.LocaleData localeData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterEnableLocale(Ektron.Cms.Localization.LocaleData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterEnableLocale(Ektron.Cms.Localization.LocaleData
  localeData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddLocale(Ektron.Cms.Localization.LocaleData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeAddLocale(Ektron.Cms.Localization.LocaleData localeData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteLocale(Ektron.Cms.Localization.LocaleData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeDeleteLocale(Ektron.Cms.Localization.LocaleData localeData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDisableLocale(Ektron.Cms.Localization.LocaleData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeDisableLocale(Ektron.Cms.Localization.LocaleData
  localeData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeEditLocale(Ektron.Cms.Localization.LocaleData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeEditLocale(Ektron.Cms.Localization.LocaleData localeData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeEnableLocale(Ektron.Cms.Localization.LocaleData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void
  OnBeforeEnableLocale(Ektron.Cms.Localization.LocaleData
  localeData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

LocalizationObjectStrategy

Namespace: `Ektron.Cms.Extensibility`

- `LocalizationObjectStrategy()`

  Summary: Constructor.

```
public LocalizationObjectStrategy()
```

- `OnAfterMarkState(Ektron.Cms.Localization.LocalizationState, Ektron.Cms.Extensibility.LocalizationObjectStrategy.LocalizationObjectEventArgs)`

```
public virtual void
  OnAfterMarkState(Ektron.Cms.Localization.LocalizationState state,
  Ektron.Cms.Extensibility.LocalizationObjectStrategy.LocalizationObjectEventArgs
  eventArgs)
```

- `OnBeforeGetContentLocalizationState (Ektron.Cms.Extensibility.LocalizationObjectStrategy.ContentLocalizationStateEventArgs)`

```
public virtual void
  OnBeforeGetContentLocalizationState(Ektron.Cms.Extensibility
  .LocalizationObjectStrategy.ContentLocalizationStateEventArgs eventArgs)
```

- `OnBeforeMarkState(Ektron.Cms.Localization.LocalizationState, Ektron.Cms.Extensibility.LocalizationObjectStrategy.LocalizationObjectCancellableEventArgs)`

```
public virtual void
  OnBeforeMarkState(Ektron.Cms.Localization.LocalizationState state,
  Ektron.Cms.Extensibility.LocalizationObjectStrategy
 .LocalizationObjectCancellableEventArgs eventArgs)
```

LocalizationStrategy

Namespace: `Ektron.Cms.Extensibility`

- `LocalizationStrategy()`

  Summary: Constructor. Initializes a new instance of the LocalizationStrategy class.

```
public LocalizationStrategy()
```

- `OnAfterCompressFiles (Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs)`

  Summary: This method is called after files are copied and compressed for each target locale.

```
public virtual void OnAfterCompressFiles(
  Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs
  eventArgs)
```

- `OnAfterExport (Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs)`

  Summary: Always called when the export job is done, whether cancelled or not. The job id is provided in the eventArgs.

```
public virtual void OnAfterExport(
  Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs
  eventArgs)
```

- `OnAfterExportItem(Ektron.Cms.Localization.LocalizableItem, Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs)`

  Summary: This method is called after a CMS object is exported.

```
public virtual void OnAfterExportItem(
  Ektron.Cms.Localization.LocalizableItem item,
  Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs
  eventArgs)
```

- `OnAfterImport (Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs)`

  Summary: Always called when the import job is done, whether cancelled or not. The job id is provided in the eventArgs.

```
public virtual void OnAfterImport(
  Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs|
  eventArgs)
```

- `OnAfterMergeXliff(string, ref string, Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs)`

  Summary: This method is called after imported XLIFF is merged with its skeleton.

```
public virtual void OnAfterMergeXliff(
  string xliffFilePath, ref string translatedContent,
  Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs
  eventArgs)
```

- `OnAfterUpdate(string, string, int, int, System.DateTime, Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs)`

Summary: This method is called after the content is updated in the database. ObjectType and ObjectId are provided in the eventArgs.

```
public virtual void OnAfterUpdate(
  string fileName, string skeletonFileName,
  int translatedLanguageID,
  int originalLanguageID,
  System.DateTime dateModified,
  Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs
  eventArgs)
```

- `OnAfterValidateXliff(string, Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs)`

  Summary: This method is called after imported XLIFF is validated.

```
public virtual void
  OnAfterValidateXliff(string xliffFilePath,
  Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs
  eventArgs)
```

- `OnAfterZipFiles(System.Collections.Generic.List<string>, System.Collections.Generic.List<string>, System.Collections.Generic.Dictionary<string,System.Collections.Generic.List<string>>, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after files are zipped together.

```
public virtual void
  OnAfterZipFiles(System.Collections.Generic.List<string> fileNames,
  System.Collections.Generic.List<string> zipFileNames,
  System.Collections.Generic.Dictionary<string,List<string>>
  filesPerZip, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeCompressFiles(int, System.Collections.Generic.List<Ektron.Cms.Localization.LocaleData>, string, ref string, ref long, Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs)`

  Summary: This method is called before files are copied and compressed for each target locale. May be cancelled. OnAfterCompressFiles is always called, even if cancelled.

```
public virtual void
  OnBeforeCompressFiles(int sourceLanguageId,
  System.Collections.Generic.List<LocaleData> targetLocales,
  string path, ref string xliffFileNameFormat,
  ref long maxZipFileSize,
  Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs
  eventArgs)
```

- `OnBeforeExport(Ektron.Cms.Localization.LocalizationExportJob, string, Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs)`

Summary: This method is called before a XLIFF export occurs. May be cancelled. OnAfterExport is always called, even if cancelled.

```
public virtual void
  OnBeforeExport(Ektron.Cms.Localization.LocalizationExportJob exportJob,
  string xliffPath,
  Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs
  eventArgs)
```

- `OnBeforeExportItem(Ektron.Cms.Localization.LocalizableItem, Ektron.Cms.Localization.LocalizationState, Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs)`

  Summary: This method is called before a CMS object is exported.

```
public virtual void
  OnBeforeExportItem(Ektron.Cms.Localization.LocalizableItem item,
  Ektron.Cms.Localization.LocalizationState locState,
  Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs
  eventArgs)
```

- `OnBeforeImport(string, Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs)`

  Summary: This method is called before a XLIFF import occurs. May be cancelled. OnAfterImport is always called, even if cancelled.

```
public virtual void
  OnBeforeImport(string xliffUrl,
  Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs
  eventArgs)
```

- `OnBeforeMergeXliff(string, string, Ektron.Cms.Xslt.ArgumentList, System.Xml.XmlUrlResolver, Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs)`

  Summary: This method is called before imported XLIFF is merged with its skeleton.

```
public virtual void
  OnBeforeMergeXliff(string xliffFilePath, string mergeXsltFilePath,
  Ektron.Cms.Xslt.ArgumentList args, System.Xml.XmlUrlResolver
  skeletonResolver,
  Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs
  eventArgs)
```

- `OnBeforeUpdate(string, string, int, int, System.DateTime, Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs)`

  Summary: This method is called before the content is updated in the database.

```
public virtual void
  OnBeforeUpdate(string fileName, string skeletonFileName,
  int translatedLanguageID, int originalLanguageID,
  System.DateTime dateModified,
```

```
  Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs
  eventArgs)
```

- `OnBeforeValidateXliff(string, Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs)`

  Summary: This method is called before imported XLIFF is validated.

```
public virtual void
  OnBeforeValidateXliff(string xliffFilePath,
  Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs
  eventArgs)
```

- `OnBeforeZipFiles(string, ref string[], ref string, ref string, ref long, Ektron.Cms.Extensibility.CancellableEventArgs)`

  Summary: This method is called before files are zipped together.

```
public virtual void
  OnBeforeZipFiles(string path, ref string[] fileNamePatterns,
  ref string zipFilePath, ref string zipFileNameFormat,
  ref long maxZipFileSize, Ektron.Cms.Extensibility.CancellableEventArgs
  eventArgs)
```

- `OnCreateSkeleton(ref string, ref string, Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs)`

  Summary: This method is called when an XLIFF skeleton is created, but before it is saved. ObjectType and ObjectId are provided in the eventArgs.

```
public virtual void
  OnCreateSkeleton(ref string fileName, ref string skeleton,
  Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs
  eventArgs)
```

- `OnCreateXliff(ref string, ref string, Ektron.Cms.Extensibility.LocalizationStrategy.CreateXliffEventArgs)`

  Summary: This method is called when an XLIFF file is created from a skeleton, but before it is saved. One XLIFF file is created for all target languages.

```
public virtual void
  OnCreateXliff(ref string fileName, ref string xliff,
  Ektron.Cms.Extensibility.LocalizationStrategy.CreateXliffEventArgs
  eventArgs)
```

- `OnGetExportList(Ektron.Cms.Localization.LocalizationExportJob, System.Collections.Generic.List<Ektron.Cms.Localization.LocalizableItem>, Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs)`

  Summary: This method is called when the list of items to export is generated. To export more items, add them to the list. To prevent exporting items, remove them from the list.

```
public virtual void
  OnGetExportList(Ektron.Cms.Localization.LocalizationExportJob
  exportJob, System.Collections.Generic.List<LocalizableItem> itemList,
  Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs
  eventArgs)
```

- `OnGetImportList(System.Collections.Generic.List<string>, Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs)`

  Summary: This method is called when the list of files to import is generated. To import more files, add them to the list. To prevent importing files, remove them from the list.

```
public virtual void
  OnGetImportList(System.Collections.Generic.List<string> fileNames,
  Ektron.Cms.Extensibility.LocalizationStrategy.LocalizationEventArgs
  eventArgs)
```

- `OnGetUnzippedFile(string, Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs)`

  Summary: This method is called after a zip file is unzipped, and for each file present in the zip.

```
public virtual void
  OnGetUnzippedFile(string fileNameAndPath,
  Ektron.Cms.Extensibility.LocalizationStrategy.CancellableLocalizationEventArgs
  eventArgs)
```

MachineTranslationStrategy

Namespace: `Ektron.Cms.Extensibility`

- `MachineTranslationStrategy()`

  Summary: Constructor. Initializes a new instance of the MachineTranslationStrategy class.

```
public MachineTranslationStrategy()
```

- `Translate(ref string, Ektron.Cms.Extensibility.MachineTranslationStrategy.MachineTranslationEventArgs)`

  Summary: Translates the given content using machine translation. Typically `eventArgs.Locale.Tag.PrivateUseSubtag` will be `mt`.

```
public virtual void Translate(ref string content,
  Ektron.Cms.Extensibility.MachineTranslationStrategy.MachineTranslationEventArgs
  eventArgs)
```

MenuStrategy

Namespace: `Ektron.Cms.Extensibility`

- MenuStrategy()

  Summary: Constructor.

```
public MenuStrategy()
```

- OnAfterAddMenu(Ektron.Cms.Organization.MenuData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void
  OnAfterAddMenu(Ektron.Cms.Organization.MenuData menuData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterAddMenuItem(Ektron.Cms.Organization.MenuItemData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void OnAfterAddMenuItem(Ektron.Cms.Organization.MenuItemData
  menuItemData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterAddSubMenu(Ektron.Cms.Organization.SubMenuData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void OnAfterAddSubMenu(Ektron.Cms.Organization.SubMenuData
  subMenuData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterDeleteMenu(long, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void OnAfterDeleteMenu(long menuId,
 Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterDeleteMenuItem(long, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void OnAfterDeleteMenuItem(long menuItemId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterDeleteSubMenu(long, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void OnAfterDeleteSubMenu(long subMenuId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterUpdateMenu(Ektron.Cms.Organization.MenuData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void OnAfterUpdateMenu(Ektron.Cms.Organization.MenuData
  menuData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterUpdateMenuItem(Ektron.Cms.Organization.MenuItemData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void OnAfterUpdateMenuItem(
  Ektron.Cms.Organization.MenuItemData menuItemData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterUpdateSubMenu(Ektron.Cms.Organization.SubMenuData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void OnAfterUpdateSubMenu(
  Ektron.Cms.Organization.SubMenuData subMenuData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnBeforeAddMenu(Ektron.Cms.Organization.MenuData, Ektron.Cms.Extensibility.CmsEventArgs)

```
public virtual void OnBeforeAddMenu(
  Ektron.Cms.Organization.MenuData menuData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddMenuItem(Ektron.Cms.Organization.MenuItemData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeAddMenuItem(
  Ektron.Cms.Organization.MenuItemData menuItemData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddSubMenu(Ektron.Cms.Organization.SubMenuData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeAddSubMenu(
  Ektron.Cms.Organization.SubMenuData subMenuData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteMenu(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeDeleteMenu(long menuId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteMenuItem(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeDeleteMenuItem(long menuItemId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteSubMenu(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeDeleteSubMenu(long subMenuId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateMenu(Ektron.Cms.Organization.MenuData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeUpdateMenu(Ektron.Cms.Organization.MenuData
  menuData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateMenuItem(Ektron.Cms.Organization.MenuItemData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeUpdateMenuItem(
  Ektron.Cms.Organization.MenuItemData menuItemData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateSubMenu(Ektron.Cms.Organization.SubMenuData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeUpdateSubMenu(
  Ektron.Cms.Organization.SubMenuData subMenuData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

MessageBoardStrategy

Namespace: `Ektron.Cms.Extensibility`

- `MessageBoardStrategy()`

  Summary: Constructor.

```
public MessageBoardStrategy()
```

- `OnAdd(Ektron.Cms.MessageBoardData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after new Message Board item/post is added.

```
public virtual void OnAdd(Ektron.Cms.MessageBoardData messageBoardData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a MessageBoardData object is deleted.

```
public virtual void OnAfterDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterReplyAdd(Ektron.Cms.MessageBoardData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after someone replies to Message Board post.

```
public virtual void OnAfterReplyAdd(
 Ektron.Cms.MessageBoardData messageBoardData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.MessageBoardData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a MessageBoardData object is edited.

```
public virtual void OnAfterUpdate(Ektron.Cms.MessageBoardData
  messageBoardData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Message Board post is deleted.

```
public virtual void OnDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnMessageApprove(Ektron.Cms.MessageBoardData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before Message Board post is approved.

```
public virtual void OnMessageApprove(Ektron.Cms.MessageBoardData
  messageBoardData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnUpdate(Ektron.Cms.MessageBoardData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after Message Board post is updated.

```
public virtual void OnUpdate(Ektron.Cms.MessageBoardData
  messageBoardData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

MetaDataStrategy

Namespace: `Ektron.Cms.Extensibility`

Valid name: `MetaData`

NOTE: These strategies are for metadata types. For metadata values, subscribe to *ContentStrategy* on page 2576.

- `MetaDataStrategy()`

  Summary: Constructor.

```
public MetaDataStrategy()
```

- `OnAfterMetaTypeAdded(Ektron.Cms.MetaTypeBaseData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a meta type is added.

```
public virtual void OnAfterMetaTypeAdded(Ektron.Cms.MetaTypeBaseData
  metaType, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterMetaTypeDeleted(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a meta type is deleted.

```
public virtual void OnAfterMetaTypeDeleted(long metaTypeId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterMetaTypeEdited(Ektron.Cms.MetaTypeBaseData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a meta type is edited.

```
public virtual void OnAfterMetaTypeEdited(Ektron.Cms.MetaTypeBaseData
  metaType, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

MicroMessageStrategy

Namespace: `Ektron.Cms.Extensibility`

- `MicroMessageStrategy()`

  Summary: Constructor.

```
public MicroMessageStrategy()
```

- `OnAfterAdd(Ektron.Cms.MicroMessageData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after new Micro Message post is added.

```
public virtual void OnAfterAdd(Ektron.Cms.MicroMessageData
  microMessageData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Micro Message post is deleted.

```
public virtual void OnAfterDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.MicroMessageData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a MicroMessage is edited.

```
public virtual void OnAfterUpdate(Ektron.Cms.MicroMessageData
  microMessageData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAdd(Ektron.Cms.MicroMessageData, Ektron.Cms.Extensibility.CmsEventArgs)`

Summary: This method is called before a Micro Message post is added.

```
public virtual void OnBeforeAdd(Ektron.Cms.MicroMessageData
  microMessageData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a Micro Message post is deleted.

```
public virtual void OnBeforeDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdate(Ektron.Cms.MicroMessageData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeUpdate(Ektron.Cms.MicroMessageData
  microMessageData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

NotificationAgentSettingStrategy

Namespace: `Ektron.Cms.Extensibility`

- `NotificationAgentSettingStrategy()`

  Summary: Constructor.

```
public NotificationAgentSettingStrategy()
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a NotificationAgentData object is deleted.

```
public virtual void OnAfterDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.Notifications.NotificationAgentData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a NotificationAgentData object is edited.

```
public virtual void OnAfterUpdate(
  Ektron.Cms.Notifications.NotificationAgentData notificationAgentData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

NotificationPreferenceStrategy

Namespace: `Ektron.Cms.Extensibility`

- `NotificationPreferenceStrategy()`

  Summary: Constructor.

```
public NotificationPreferenceStrategy()
```

- `OnAfterDeleteNotificationPreference(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a NotificationPreference is deleted.

```
public virtual void OnAfterDeleteNotificationPreference(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateNotificationPreference (Ektron.Cms.Notifications.NotificationPreferenceData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a NotificationPreference is edited.

```
public virtual void OnAfterUpdateNotificationPreference(
  Ektron.Cms.Notifications.NotificationPreferenceData
  notificationPreferenceData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

PermissionStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterAddPermissionData(Ektron.Cms.PermissionData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterAddPermissionData(Ektron.Cms.PermissionData
  data, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDeletePermissionData(Ektron.Cms.PermissionData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterDeletePermissionData(
  Ektron.Cms.PermissionData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateItemInheritance(long, string, bool, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterUpdateItemInheritance(long itemId,
  string requestType, bool enable,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdatePermissionData(Ektron.Cms.PermissionData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterUpdatePermissionData(
  Ektron.Cms.PermissionData data, Ektron.Cms.Extensibility.CmsEventArgs
  eventArgs)
```

- `OnAfterUpdatePrivateSetting(long, string, bool, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterUpdatePrivateSetting(long itemId,
  string requestType, bool enable,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddPermissionData(Ektron.Cms.PermissionData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeAddPermissionData(
  Ektron.Cms.PermissionData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeletePermissionData(Ektron.Cms.PermissionData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeDeletePermissionData(
  Ektron.Cms.PermissionData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateItemInheritance(long, string, bool, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeUpdateItemInheritance(long itemId,
  string requestType, bool enable,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdatePermissionData(Ektron.Cms.PermissionData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeUpdatePermissionData(
  Ektron.Cms.PermissionData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdatePrivateSetting(long, string, bool, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeUpdatePrivateSetting(long itemId,
  string requestType, bool enable,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

PseudoLocalizationStrategy

Namespace: `Ektron.Cms.Extensibility`

- `PseudoLocalizationStrategy()`

  Summary: Constructor. Initializes a new instance of the PseudoLocalizationStrategy class.

```
public PseudoLocalizationStrategy()
```

- `PseudoLocalize(ref string, Ektron.Cms.Extensibility.PseudoLocalizationStrategy.PseudoLocalizationEventArgs)`

  Summary: Localizes the given content into a "pseudo" language to test translation-readiness. Typically, `eventArgs.Locale.Tag.PrivateUseSubtag` will be `pseudo`.

```
public virtual void PseudoLocalize(ref string content,
  Ektron.Cms.Extensibility.PseudoLocalizationStrategy.PseudoLocalizationEventArgs
  eventArgs)
```

QueryStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterAdvancedQuery(Ektron.Cms.Search.AdvancedSearchCriteria, Ektron.Cms.Search.SearchResponseData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Signals the completion of an 'advanced' query.

```
public virtual void OnAfterAdvancedQuery(
  Ektron.Cms.Search.AdvancedSearchCriteria criteria,
  Ektron.Cms.Search.SearchResponseData response,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterKeywordQuery(Ektron.Cms.Search.KeywordSearchCriteria, Ektron.Cms.Search.SearchResponseData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Signals the completion of a 'keyword' query.

```
public virtual void OnAfterKeywordQuery(
  Ektron.Cms.Search.KeywordSearchCriteria criteria,
  Ektron.Cms.Search.SearchResponseData response,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAdvancedQuery(Ektron.Cms.Search.AdvancedSearchCriteria, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Signals that an 'advanced' query is about to be requested.

```
public virtual void OnBeforeAdvancedQuery(
  Ektron.Cms.Search.AdvancedSearchCriteria criteria,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeKeywordQuery(Ektron.Cms.Search.KeywordSearchCriteria, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Signals that a 'keyword' query is about to be requested.

```
public virtual void OnBeforeKeywordQuery(
  Ektron.Cms.Search.KeywordSearchCriteria criteria,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `QueryStrategy()`

  Summary: Constructor.

```
public QueryStrategy()
```

RatingStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterDeleteRating(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Object Rating is deleted.

```
public virtual void OnAfterDeleteRating(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateRating(Ektron.Cms.RatingData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Object Rating is edited.

```
public virtual void OnAfterUpdateRating(Ektron.Cms.RatingData
  ratingData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `RatingStrategy()`

  Summary: Constructor.

```
public RatingStrategy()
```

RoleStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterAddCustomRolePermission(string, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterAddCustomRolePermission(string roleName,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterAddOrUpdateRoleMember(long, Ektron.Cms.RoleMemberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterAddOrUpdateRoleMember(long roleId,
  Ektron.Cms.RoleMemberData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDeleteCustomRolePermission(string, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterDeleteCustomRolePermission(string roleName,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDropRoleMember(long, Ektron.Cms.RoleMemberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterDropRoleMember(long roleId,
  Ektron.Cms.RoleMemberData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddCustomRolePermission(string, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeAddCustomRolePermission(string roleName,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddOrUpdateRoleMember(long, Ektron.Cms.RoleMemberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeAddOrUpdateRoleMember(long roleId,
  Ektron.Cms.RoleMemberData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteCustomRolePermission(string, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeDeleteCustomRolePermission(string
  roleName, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDropRoleMember(long, Ektron.Cms.RoleMemberData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeDropRoleMember(long roleId,
  Ektron.Cms.RoleMemberData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

SyncCommentEventArgs

Namespace: `Ektron.Cms.Extensibility`

- `SyncCommandEventArgs()`

  Summary: Constructor. Initializes a new instance of the SyncCommandEventArgs class.

```
public SyncCommandEventArgs()
```

- StrategyTypeRequestingCancellation

```
public System.Type StrategyTypeRequestingCancellation { set; get; }
```

- SyncCancellationArgs

```
public System.ComponentModel.CancelEventArgs
  SyncCancellationArgs { set; get; }
```

SubscriberCustomFieldStrategy

Namespace: Ektron.Cms.Extensibility.Marketing

- OnAfterAdd(Ektron.Cms.Marketing.SubscriberCustomFieldData, Ektron.Cms.Extensibility.CmsEventArgs))

  Summary: Called after an item is added.

```
public virtual void
  OnAfterAdd(Ektron.Cms.Marketing.SubscriberCustomFieldData
  subscribercustomfielddata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)

  Summary: Called after an item is deleted.

```
public virtual void
  OnAfterDelete(long id, Ektron.Cms.Extensibility.CmsEventArgs
  eventArgs)
```

- OnAfterUpdate(Ektron.Cms.SubscriberCustomFieldData, Ektron.Cms.Extensibility.CmsEventArgs)

  Summary: Called after an item is updated.

```
public virtual void
  OnAfterUpdate(Ektron.Cms.SubscriberCustomFieldData
  subscribercustomfielddata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnBeforeAdd(Ektron.Cms.SubscriberCustomFieldData, Ektron.Cms.Extensibility.CmsEventArgs)

  Summary: Called before an item is added.

```
public virtual void
  OnBeforeAdd(Ektron.Cms.SubscriberCustomFieldData
  subscribercustomfielddata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- OnBeforeDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)

  Summary: Called before a list item is deleted.

```
public virtual void
  OnBeforeDelete(long id, Ektron.Cms.Extensibility.CmsEventArgs
  eventArgs)
```

- OnBeforeUpdate(Ektron.Cms.SubscriberCustomFieldData, Ektron.Cms.Extensibility.CmsEventArgs)

  Summary: Called before a list item is updated.

```
public virtual void
  OnBeforeUpdate(Ektron.Cms.SubscriberCustomFieldData
  subscribercustomfielddata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `SubscriberCustomFieldStrategy()`

  Summary: Constructor

```
public SubscriberCustomFieldStrategy()
```

SubscriberStrategy

Namespace: `Ektron.Cms.Extensibility.Marketing`

- `OnAfterAdd(Ektron.Cms.Marketing.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs))`

  Summary: Called after an item is added.

```
public virtual void
  OnAfterAdd(Ektron.Cms.Marketing.SubscriberData
  subscriberdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called after an item is deleted.

```
public virtual void
  OnAfterDelete(long id, Ektron.Cms.Extensibility.CmsEventArgs
  eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called after an item is updated.

```
public virtual void
  OnAfterUpdate(Ektron.Cms.SubscriberData
  subscriberdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAdd(Ektron.Cms.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called before an item is added.

```
public virtual void
  OnBeforeAdd(Ektron.Cms.SubscriberData
  subscriberdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Called before a list item is deleted.

```
public virtual void
  OnBeforeDelete(long id, Ektron.Cms.Extensibility.CmsEventArgs
  eventArgs)
```

- `OnBeforeUpdate(Ektron.Cms.SubscriberData, Ektron.Cms.Extensibility.CmsEventArgs)`

Summary: Called before a list item is updated.

```
public virtual void
  OnBeforeUpdate(Ektron.Cms.SubscriberData
  subscriberdata,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `SubscriberStrategy()`

  Summary: Constructor

```
public SubscriberStrategy()
```

TagStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAdd(Ektron.Cms.TagData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after Tag is added.

```
public virtual void OnAdd(Ektron.Cms.TagData tagData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterObjectTagDeleted(long, long, EkEnumeration.CMSObjectTypes, long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterObjectTagDeleted(long tagId, long objectId,
  EkEnumeration.CMSObjectTypes objectType, long userId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterObjectTagged(Ektron.Cms.TagAssignmentData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterObjectTagged(Ektron.Cms.TagAssignmentData
  tagAssignment, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Tag is deleted.

```
public virtual void OnDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnUpdate(Ektron.Cms.TagData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Tag is updated.

```
public virtual void OnUpdate(Ektron.Cms.TagData tagData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `TagStrategy()`

  Summary: Constructor.

```
public TagStrategy()
```

TaskCategoryStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterDeleteTaskCategory(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a task category is deleted.

```
public virtual void OnAfterDeleteTaskCategory(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateTaskCategory(Ektron.Cms.TaskCategoryData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a task category is edited.

```
public virtual void OnAfterUpdateTaskCategory(
  Ektron.Cms.TaskCategoryData taskCategoryData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `TaskCategoryStrategy()`

  Summary: Constructor.

```
public TaskCategoryStrategy()
```

TaskStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterDeleteTask(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a task is deleted.

```
public virtual void OnAfterDeleteTask(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateTask(Ektron.Cms.TaskData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a task is edited.

```
public virtual void OnAfterUpdateTask(
  Ektron.Cms.TaskData taskData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `TaskStrategy()`

  Summary: Constructor.

```
public TaskStrategy()
```

TaxonomyItemStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterDeleteTaxonomyItem(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a TaxonomyItem is deleted.

```
public virtual void OnAfterDeleteTaxonomyItem(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateTaxonomyItem(Ektron.Cms.TaxonomyItemData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a TaxonomyItem is edited.

```
public virtual void OnAfterUpdateTaxonomyItem(
  Ektron.Cms.TaxonomyItemData taxonomyItem,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `TaxonomyItemStrategy()`

  Summary: Constructor.

```
public TaxonomyItemStrategy()
```

TaxonomyStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterAdd(Ektron.Cms.TaxonomyData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a new Taxonomy is added.

```
public virtual void OnAfterAdd(Ektron.Cms.TaxonomyData taxonomyData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterAssignItem(Ektron.Cms.TaxonomyRequest, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Taxonomy Item is added/assigned to Taxonomy.

```
public virtual void OnAfterAssignItem(Ektron.Cms.TaxonomyRequest
  request, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterAssignItem(Ektron.Cms.TaxonomySyncRequest, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterAssignItem(Ektron.Cms.TaxonomySyncRequest
  request, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterCopy(long, long, int, bool, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after an item is added to a taxonomy node.

```
public virtual void OnAfterCopy(long sourceId, long newId,
  int languageId, bool deleteSource,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after Taxonomy is deleted.

```
public virtual void OnAfterDelete(long id, int languageId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterRemoveItem(Ektron.Cms.TaxonomyRequest, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after an item is removed from a taxonomy node.

```
public virtual void OnAfterRemoveItem(Ektron.Cms.TaxonomyRequest
  request, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterRemoveItem(Ektron.Cms.TaxonomySyncRequest, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after an item is removed from a taxonomy node.

```
public virtual void OnAfterRemoveItem(Ektron.Cms.TaxonomySyncRequest
  request, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.TaxonomyData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Taxonomy is updated.

```
public virtual void OnAfterUpdate(Ektron.Cms.TaxonomyData taxonomyData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAdd(Ektron.Cms.TaxonomyData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a new Taxonomy is added.

```
public virtual void OnBeforeAdd(Ektron.Cms.TaxonomyData taxonomyData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAssignItem(Ektron.Cms.TaxonomyRequest, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before Taxonomy item is added/assigned to Taxonomy.

```
public virtual void OnBeforeAssignItem(Ektron.Cms.TaxonomyRequest
  request, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAssignItem(Ektron.Cms.TaxonomySyncRequest, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before an item is added to a taxonomy node.

```
public virtual void OnBeforeAssignItem(Ektron.Cms.TaxonomySyncRequest
  request, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeCopy(long, long, int, bool, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a taxonomy node is copied.

```
public virtual void OnBeforeCopy(long sourceId, long destinationId,
  int languageId, bool deleteSource,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a Taxonomy is deleted.

```
public virtual void OnBeforeDelete(long id, int languageId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeRemoveItem(Ektron.Cms.TaxonomyRequest, Ektron.Cms.Extensibility.CmsEventArgs)`

Summary: This method is called before an item is removed from a taxonomy node.

```
public virtual void OnBeforeRemoveItem(Ektron.Cms.TaxonomyRequest
  request, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeRemoveItem(Ektron.Cms.TaxonomySyncRequest, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before an item is removed from a taxonomy node.

```
public virtual void OnBeforeRemoveItem(Ektron.Cms.TaxonomySyncRequest
  request, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdate(Ektron.Cms.TaxonomyData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a Taxonomy is updated.

```
public virtual void OnBeforeUpdate(Ektron.Cms.TaxonomyData
  taxonomyData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `TaxonomyStrategy()`

  Summary: Constructor.

```
public TaxonomyStrategy()
```

TemplateStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterAddTemplate(Ektron.Cms.TemplateData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterAddTemplate(Ektron.Cms.TemplateData
  TemplateData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDeleteTemplate(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterDeleteTemplate(long Id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateTemplate(Ektron.Cms.TemplateData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnAfterUpdateTemplate(
 Ektron.Cms.TemplateData TemplateData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddTemplate(Ektron.Cms.TemplateData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeAddTemplate(Ektron.Cms.TemplateData
  TemplateData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteTemplate(long, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeDeleteTemplate(long Id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateTemplate(Ektron.Cms.TemplateData, Ektron.Cms.Extensibility.CmsEventArgs)`

```
public virtual void OnBeforeUpdateTemplate(Ektron.Cms.TemplateData
  TemplateData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `TemplateStrategy()`

  Summary: Constructor.

```
public TemplateStrategy()
```

TodoListStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterDeleteTodoList(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a TodoList is deleted.

```
public virtual void OnAfterDeleteTodoList(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateTodoList(Ektron.Cms.ToDo.TodoListData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a TodoList is edited.

```
public virtual void OnAfterUpdateTodoList(Ektron.Cms.ToDo.TodoListData
  todoList, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `TodoListStrategy()`

  Summary: Constructor.

```
public TodoListStrategy()
```

UserCustomPropertyStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterAddUserCustomProperty(Ektron.Cms.UserCustomPropertyData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user custom property is added.

```
public virtual void OnAfterAddUserCustomProperty(
  Ektron.Cms.UserCustomPropertyData userCustomPropertyData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDeleteUserCustomProperty(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user custom property is deleted.

```
public virtual void OnAfterDeleteUserCustomProperty(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateUserCustomProperty (Ektron.Cms.UserCustomPropertyData, Ektron.Cms.Extensibility.CmsEventArgs)`

Summary: This method is called after a user custom property is edited.

```
public virtual void OnAfterUpdateUserCustomProperty(
  Ektron.Cms.UserCustomPropertyData userCustomPropertyData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddUserCustomProperty(Ektron.Cms.UserCustomPropertyData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a user custom property is added.

```
public virtual void OnBeforeAddUserCustomProperty(
  Ektron.Cms.UserCustomPropertyData userCustomPropertyData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteUserCustomProperty(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user custom property is deleted.

```
public virtual void OnBeforeDeleteUserCustomProperty(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateUserCustomProperty (Ektron.Cms.UserCustomPropertyData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a user custom property is edited.

```
public virtual void OnBeforeUpdateUserCustomProperty(
  Ektron.Cms.UserCustomPropertyData userCustomPropertyData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `UserCustomPropertyStrategy()`

  Summary: Constructor.

```
public UserCustomPropertyStrategy()
```

UserGroupStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterAddUserGroup(Ektron.Cms.UserGroupData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user group is added.

```
public virtual void OnAfterAddUserGroup(Ektron.Cms.UserGroupData
  userGroupData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDeleteUserGroup(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user group is deleted.

```
public virtual void OnAfterDeleteUserGroup(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateUserGroup(Ektron.Cms.UserGroupData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user group is edited.

```
public virtual void OnAfterUpdateUserGroup(Ektron.Cms.UserGroupData
  userGroupData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `UserGroupStrategy()`

  Summary: Constructor.

```
public UserGroupStrategy()
```

UserNotificationSettingStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a UserNotificationSettingData object is deleted.

```
public virtual void OnAfterDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate (Ektron.Cms.Notifications.UserNotificationSettingData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a UserNotificationSettingData object is edited.

```
public virtual void OnAfterUpdate(
  Ektron.Cms.Notifications.UserNotificationSettingData
  userNotificationSettingData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `UserNotificationSettingStrategy()`

  Summary: Constructor.

```
public UserNotificationSettingStrategy()
```

UserStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterAddColleague(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a colleague/friend is added.

```
public virtual void OnAfterAddColleague(long userOneId, long userTwoId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterAddUser(Ektron.Cms.UserData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a new user is added.

```
public virtual void OnAfterAddUser(Ektron.Cms.UserData userData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterColleagueRequest(Ektron.Cms.Community.ActionRequestData, Ektron.Cms.Extensibility.CmsEventArgs)`

Summary: This method is called after a colleague/friend request is sent.

```
public virtual void OnAfterColleagueRequest(
  Ektron.Cms.Community.ActionRequestData invitationData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDeleteColleague(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after two users' colleague relationship is deleted. There is no significance to the ordering of the users.

```
public virtual void OnAfterDeleteColleague(long userOneId,
  long userTwoId, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDeleteUser(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user is deleted.

```
public virtual void OnAfterDeleteUser(long Id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterLogin(Ektron.Cms.UserData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user logs in.

```
public virtual void OnAfterLogin(Ektron.Cms.UserData userData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterLogout(Ektron.Cms.UserData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user is logged out of the CMS.

```
public virtual void OnAfterLogout(Ektron.Cms.UserData userData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateUser(Ektron.Cms.UserData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user is updated.

```
public virtual void OnAfterUpdateUser(Ektron.Cms.UserData userData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUserAddInGroup(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user added in the group.

```
public virtual void OnAfterUserAddInGroup(long userId, long groupId,
 Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUserDeleteFromGroup(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a user deleted from group.

```
public virtual void OnAfterUserDeleteFromGroup(long userId,
  long groupId, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddUser(Ektron.Cms.UserData, Ektron.Cms.Extensibility.CmsEventArgs)`

Summary: This method is called before a new user is added to CMS.

```
public virtual void OnBeforeAddUser(Ektron.Cms.UserData userData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDeleteUser(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a user is deleted from CMS.

```
public virtual void OnBeforeDeleteUser(long Id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeLogin(Ektron.Cms.UserData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a user logs in.

```
public virtual void OnBeforeLogin(Ektron.Cms.UserData userData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeLogout(Ektron.Cms.UserData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a user is logged out of the CMS.

```
public virtual void OnBeforeLogout(Ektron.Cms.UserData userData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdateUser(Ektron.Cms.UserData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a user is updated.

```
public virtual void OnBeforeUpdateUser(Ektron.Cms.UserData userData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUserAddInGroup(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a user add in to the group.

```
public virtual void OnBeforeUserAddInGroup(long userId, long groupId,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUserDeleteFromGroup(long, long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a user delete from group.

```
public virtual void OnBeforeUserDeleteFromGroup(long userId,
  long groupId, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `UserStrategy()`

  Summary: Constructor.

```
public UserStrategy()
```

WebEventStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterAdd(Ektron.Cms.Common.Calendar.WebEventData, Ektron.Cms.Extensibility.CmsEventArgs)`

Summary: This method is called after a new Web calendar event is added.

```
public virtual void OnAfterAdd(Ektron.Cms.Common.Calendar.WebEventData
  eventData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterAddVariance(Ektron.Cms.Common.Calendar.WebEventData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Web calendar event variance is added.

```
public virtual void OnAfterAddVariance(
  Ektron.Cms.Common.Calendar.WebEventData
  eventData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterCancelOccurrence(Ektron.Cms.Common.Calendar.WebEventData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a single occurrence of Web calendar event (recursive event) is cancelled.

```
public virtual void OnAfterCancelOccurrence(
  Ektron.Cms.Common.Calendar.WebEventData eventData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Web calendar event is deleted.

```
public virtual void OnAfterDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterPublish(Ektron.Cms.Common.Calendar.WebEventData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after publishing a webevent.

```
public virtual void OnAfterPublish(
  Ektron.Cms.Common.Calendar.WebEventData eventData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdate(Ektron.Cms.Common.Calendar.WebEventData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called after a Web calendar event is updated.

```
public virtual void OnAfterUpdate(
  Ektron.Cms.Common.Calendar.WebEventData eventData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAdd(Ektron.Cms.Common.Calendar.WebEventData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a new Web calendar event is added.

```
public virtual void OnBeforeAdd(
  Ektron.Cms.Common.Calendar.WebEventData eventData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeAddVariance(Ektron.Cms.Common.Calendar.WebEventData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a Web calendar event variance is added.

```
public virtual void OnBeforeAddVariance(
  Ektron.Cms.Common.Calendar.WebEventData eventData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeCancelOccurrence(Ektron.Cms.Common.Calendar.WebEventData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a single occurrence of a Web calendar event (recursive event) is canceled.

```
public virtual void OnBeforeCancelOccurrence(
  Ektron.Cms.Common.Calendar.WebEventData eventData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeDelete(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a Web calendar event is deleted.

```
public virtual void OnBeforeDelete(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforePublish(Ektron.Cms.Common.Calendar.WebEventData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a Web calendar event is published.

```
public virtual void OnBeforePublish(
  Ektron.Cms.Common.Calendar.WebEventData eventData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnBeforeUpdate(Ektron.Cms.Common.Calendar.WebEventData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: This method is called before a Web calendar event is updated.

```
public virtual void OnBeforeUpdate(
 Ektron.Cms.Common.Calendar.WebEventData eventData,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `WebEventStrategy()`

  Summary: Constructor.

```
public WebEventStrategy()
```

XmlConfigurationStrategy

Namespace: `Ektron.Cms.Extensibility`

- `OnAfterDeleteXmlConfiguration(long, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Executes a strategy method after an XML configuration was deleted.

```
public virtual void OnAfterDeleteXmlConfiguration(long id,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterPublishXmlConfiguration(XmlConfigData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Executes a strategy method after publishing an XML configuration.

```
public virtual void OnAfterPublishXmlConfiguration(XmlConfigData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateXmlConfiguration(XmlConfigData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Executes a strategy method after an XML configuration update.

```
public virtual void OnAfterUpdateXmlConfiguration(XmlConfigData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

- `OnAfterUpdateXmlIndex(XmlConfigData, Ektron.Cms.Extensibility.CmsEventArgs)`

  Summary: Executes a strategy method after updating index fields (2nd screen) while updating smart forms.

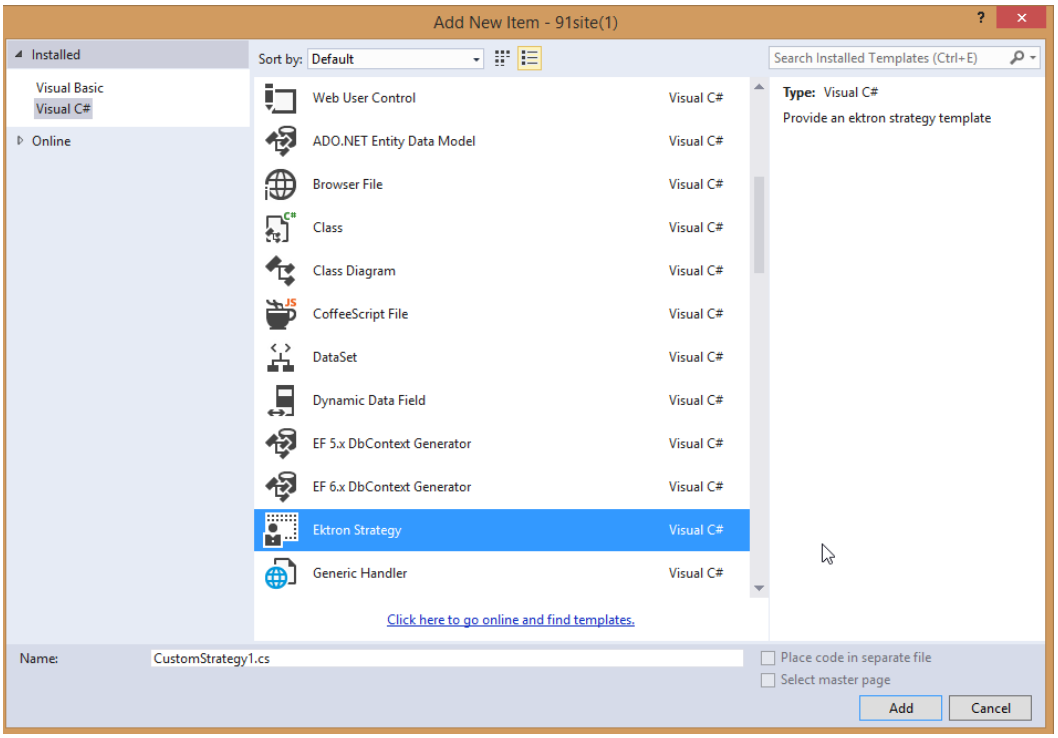
```
public virtual void OnAfterUpdateXmlIndex(XmlConfigData data,
  Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
```

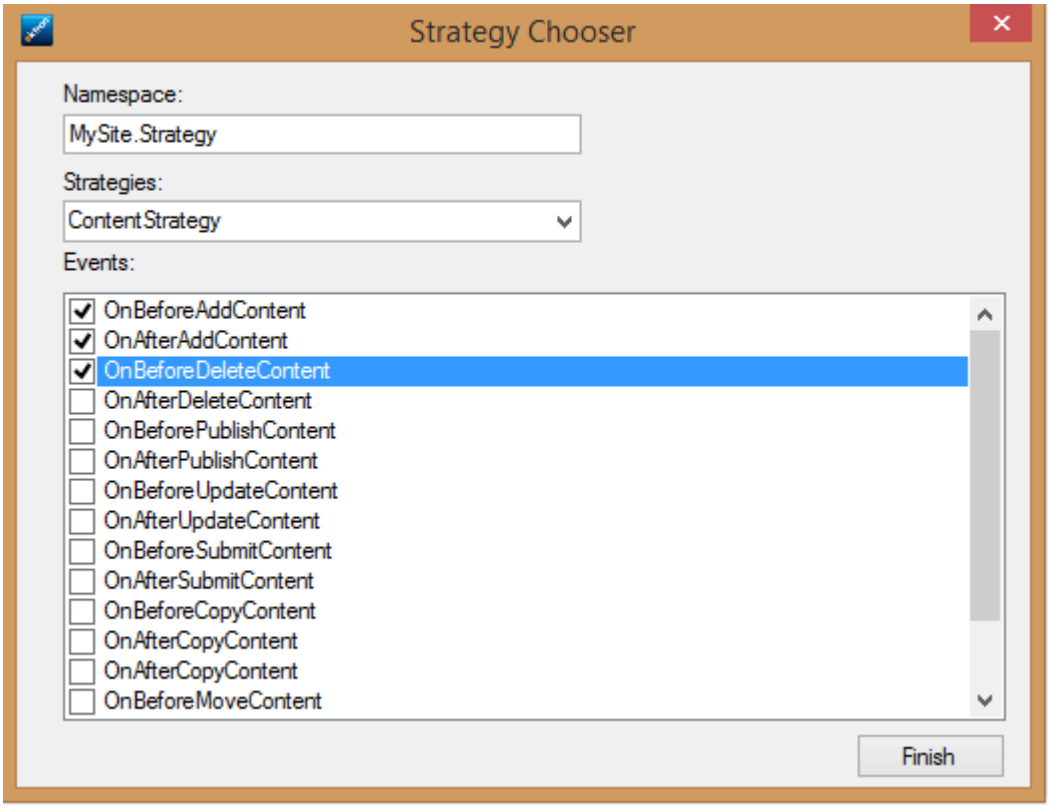
Programmatically canceling strategies

Strategies called before actions take place in the system, such as `BasketStrategy.OnBeforeBasketDelete`, are provided CancellableEventArgs instead of CmsEventArgs. By setting the `CancellableEventArgs.IsCancelled` property to **true**, the current action is not performed, and the message you specify in `CancellableEventArgs.CancellationMessage` is returned to the user.

Wizard to generate Ektron event handlers in Visual Studio

By installing the `Ektron.Addin.Strategy` extension into Visual Studio, you can use a wizard to choose Ektron Strategies to subscribe to. When finished, the wizard creates a strategy skeleton subscribed to the selected events and adds it to the `objectFactory.config` file so the site uses it.


Add New Item - 91site(1)
Installed
Visual Basic
Visual C#
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
Web User Control
ADO.NET Entity Data Model
Browser File
Class
Class Diagram
CoffeeScript File
DataSet
Dynamic Data Field
EF 5.x DbContext Generator
EF 6.x DbContext Generator
Ektron Strategy
Generic Handler
Visual C#
Type: Visual C#
Provide an ektron strategy template
Click here to go online and find templates.
Name: CustomStrategy1.cs
Place code in separate file
Select master page
Add
Cancel


Strategy Chooser
Namespace:
MySite.Strategy
Strategies:
ContentStrategy
Events:
OnBeforeAddContent
OnAfterAddContent
OnBeforeDeleteContent
OnAfterDeleteContent
OnBeforePublishContent
OnAfterPublishContent
OnBeforeUpdateContent
OnAfterUpdateContent
OnBeforeSubmitContent
OnAfterSubmitContent
OnBeforeCopyContent
OnAfterCopyContent
OnAfterCopyContent
OnBeforeMoveContent
Finish

```
namespace MySite.Strategy
{
    /// <summary>
    /// CustomStrategy1
    /// </summary>
    -references
    public class CustomStrategy1 : Ektron.Cms.Extensibility.ContentStrategy
    {
        -references
        public override void OnBeforeAddContent(Ektron.Cms.ContentData contentData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
        {
        }

        -references
        public override void OnAfterAddContent(Ektron.Cms.ContentData contentData, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
        {
        }

        -references
        public override void OnBeforeDeleteContent(System.Int64 id, Ektron.Cms.Extensibility.CmsEventArgs eventArgs)
        {
        }


    }
}
```

6

User and content constants

You can use the following information to determine constant values when working with Ektron users and content.

User constants

- Builtin: 999999999
- Internal Admin: 18611864
- Image: 1
- File: 2
- AllMemberShip Group: 888888
- Everyone Group: 2
- Admin Group: 1

Content constants

- Archive_ManagedAsset_Min: 1100
- Archive_ManagedAsset_Max: 2099
- CMSContentType_AllTypes: -1
- CMSContentType_Archive_Content: 3
- CMSContentType_Archive_Forms: 4
- CMSContentType_Archive_Media: 12
- CMSContentType_Asset: 103
- CMSContentType_BlogComments: 13
- CMSContentType_Content: 1
- CMSContentType_CatalogEntry: 3333
- CMSContentType_DiscussionTopic: 1111
- CMSContentType_Forms: 2
- CMSContentType_Image: 106
- CMSContentType_Library: 7
- CMSContentType_Media: 104
- CMSContentType_NonImageLibrary: 9
- CMSContentType_NonLibraryForms: 98
- CMSContentType_NonLibraryContent: 99
- CMSContentType_OfficeDoc: 101
- CMSContentType_PDF: 102
- CMSContentType_XmlConfig: 14
- ManagedAsset_Max: 1099
- ManagedAsset_Min: 100
- MaxNumManagedAssetTypes: 1000

7

Ektron Markup Language

The Ektron Markup Language (EkML) makes it easy for Web developers to manage the output presentation of server controls. EkML uses a simple markup that resembles HTML in syntax. The EkML is cached and .NET watches for file changes.

The `MarkupLanguage` property in the Ektron server controls and Ektron Dreamweaver functions is used to select the template file that contains the Ektron Markup Language. This language is made up of tags and variables that assign formatting information to a control or function when displayed on a Web page.

EkML templates

Ektron provides an EkML template for many server controls. The templates are located in *siteroot*`/Workarea/Templates`. At the top of each template is a list of variables that can be used with that control.

The default file follows this pattern: `\`*siteroot*`\Workarea\template\`*this object*`\`*objectname*`.ekml`. For example, `\siteroot\Workarea\template\collection\collection.ekml`.

To customize the default .ekml file, copy it to a folder outside the `siteroot\workarea` folder and edit it. Next, in a server control's `MarkupLanguage` property, enter the path to that file relative to the site root folder.

collection.ekml

This file defines which items and information are included when displaying content item information in a collection using the *Collection* on page 2209 server control.

collection.ekml variables

- *[$CollectionDescription]* on page 2657. The collection's description.
- *[$CollectionTitle]* on page 2657. The collection's title.
- *[$Comment]* on page 2658. The content's comment information.
- *[$ContentId]* on page 2659. The content item's ID.
- *[$DateCreated]* on page 2659. The date the content was created.
- *[$DateModified]* on page 2660. The date the content was last modified.
- *[$EditorFirstName]* on page 2662. The last editor's first name for a content item.
- *[$EditorLastName]* on page 2662. The last editor's last name for a content item.
- *[$FolderId]* on page 2665. The folder ID of a content item.
- *[$Html]* on page 2666. The HTML contained in the content item.
- *[$HyperLink]* on page 2666. Adds a hyperlink using the title of the content block as the text.
- *[$Image]* on page 2667. The path of the image defined in a content item's Metadata. When wrapped in `<img src=""/>` tag, the image appears. For example: `<img src="[$Image]"/>`
- *[$ImageIcon]* on page 2667. An image icon for the content item. For example, if the content item is HTML, the () icon appears.

- *[$ImageThumbnail]* on page 2667. The path for the image's thumbnail defined in a content item's Metadata. When wrapped in `<img src=""/>` tag, a thumbnail version of the image appears. For example: `<img src="[$ImageThumbnail]"/>`. See:
- *[$Index]* on page 2668. Serialize the content items in a numbered list.
- *[$ItemCount]* on page 2668. Total number of items in a list.
- *[$Language]* on page 2668. The language ID for the content item.
- *[$LinkTarget]* on page 2669. When added to an `<a href="">` tag's `target=""` attribute, this variable reads the server control's `LinkTarget` property and uses its setting.
- *[$QuickLink]* on page 2672. The Quicklink information for the content item. When wrapped in an `<a href="">` tag, you can create a Hyperlink.
- *[$SERVER_NAME]* on page 2675. The server name. For example, If this variable is applied to `http://www.example.com/demo.aspx`, the return is `www.example.com`.
- *[$ShowBubble(width,height)]* on page 2677. Similar to `[$ShowBubble]`. It calls the `<ekbubbleinfo>` tags and lets you set the width of the bubble.
- *[$ShowBubble]* on page 2676. Calls the `<ekbubbleinfo>` tags and places the information contained within those tags in a pop-up bubble.
- *[$ShowContent('htmltagid')]* on page 2677. Calls the `<ekcontentinfo>` tags and places the information in those tags within the specified HTML tag ID. Replace the 'htmltagid' with the ID of the tag.
- *[$Status]* on page 2678. A content item's status.
- *[$Teaser]* on page 2678. The content item's summary information. If the item is an HTML form, this variable is *not* supported with this ekml file.
- *[$Title]* on page 2679. The content item's title.
- *[$UrlEncode('str')]* on page 2679. Encodes string information. This variable can be used to encode another EkML variable and place it in an email.
- *[$UrlParam('paramname')]* on page 2680. The value of a query string's parameter. For example, if the query string is `?id=27` and the variable is `[$UrlParam('id')]`, 27 appears.

contentlist.ekml

This file defines which items and information are included when displaying content item information in a content list using the *ContentList* on page 2336 server control.

listsummary.ekml

This file defines which items and information are included when displaying content item information in a list summary using the *ListSummary* on page 2451 server control.

ListSummary.ekml variables

- *[$Comment]* on page 2658. The content's comment information.
- *[$ContentId]* on page 2659. The content item's ID.
- *[$DateCreated]* on page 2659. The date the content was created.
- *[$DateModified]* on page 2660. The date the content was last modified.
- *[$EditorFirstName]* on page 2662. The last editor's first name for a content item.
- *[$EditorLastName]* on page 2662. The last editor's last name for a content item.
- *[$FolderDescription]* on page 2664. The folder's description.
- *[$FolderId]* on page 2665. The folder ID of a content item.
- *[$FolderName]* on page 2665. The folder's name.
- *[$Html]* on page 2666. The HTML contained in the content item.
- *[$HyperLink]* on page 2666. Adds a hyperlink using the title of the content block as the text.
- *[$Image]* on page 2667. The path for the image defined in a content item's Metadata. When wrapped in `<img src=""/>` tag, the image appears. For example: `<img src="[$Image]"/>`
- *[$ImageIcon]* on page 2667. An image icon for the content item. For example, if the content item is HTML, the () icon appears.
- *[$ImageThumbnail]* on page 2667. The path for the image's thumbnail defined in a content item's Metadata. When wrapped in `<img src=""/>` tag, a thumbnail version of the image appears. For example: `<img src="[$ImageThumbnail]"/>`
- *[$Index]* on page 2668. Serialize the content items in a numbered list.
- *[$ItemCount]* on page 2668. The total number of items in a list.
- *[$Language]* on page 2668. The language ID for the content item.
- *[$LinkTarget]* on page 2669. When added to an `<a href="">` tag's `target=""` attribute, this variable reads the server control's `LinkTarget` property and uses its setting.
- *[$QuickLink]* on page 2672. Quicklink information for the content item. When wrapped in an `<a href="">` tag, you can create a Hyperlink.
- *[$SERVER_NAME]* on page 2675. The server name. For example, If this variable is applied to `http://www.example.com/demo.aspx`, the return is `www.example.com`.
- *[$ShowBubble(width,height)]* on page 2677. Similar to `[$ShowBubble]`. It calls the `<ekbubbleinfo>` tags and lets you set the width of the bubble.
- *[$ShowBubble]* on page 2676. Calls `<ekbubbleinfo>` tags and places the information contained within those tags in a pop-up bubble.
- *[$ShowContent('htmltagid')]* on page 2677. Calls the `<ekcontentinfo>` tags and places information in those tags within the specified HTML tag ID. Replace the 'htmltagid' with the ID of the tag.
- *[$Status]* on page 2678. The status of a content item.
- *[$Teaser]* on page 2678. The content item's summary information. If the item is an HTML form, this variable is *not* supported with this ekml file.

- *[$Title]* on page 2679 the content item's title.
- *[$UrlEncode('str')]* on page 2679. Encodes the string information. This variable can be used to encode another EkML variable and place it in an email.
- *[$UrlParam('paramname')]* on page 2680. The value of a QueryString's parameter. For example, if the QueryString is `?id=27` and the variable is `[$UrlParam('id')]`, 27 appears.

map.ekml

The `map.ekml` file defines which items and information are included when using the *Map* on page 2463 server control. Unlike most other .ekml files, the `map.emkl` has some variables that cannot be changed or moved around.

The file's variables appear in 3 <tr></tr> table rows. These rows are located below the main table. In the first 2 table rows, you can modify a tag's style information only. In the third, you can change the style information and the order of the variables to create different layouts for your page. It is recommended you hide the first 2 table rows if you are not changing their style information. For example:

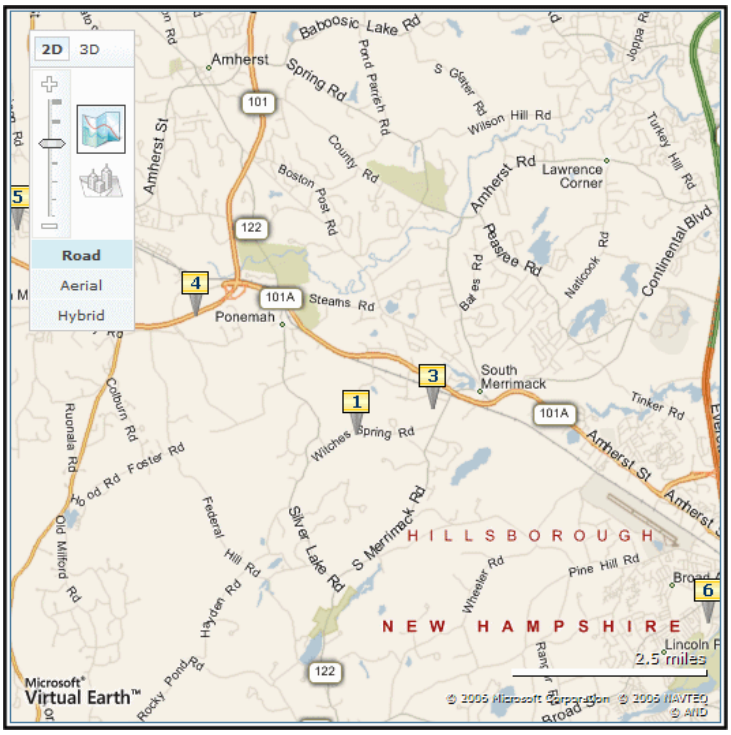
```
<ekmarkup>
  <ekoutput>
    <div>
      <table>
        <tr>...
        <tr>...
        <tr>
          <td>
            <table>
              <tr>
                <td>
                  <div id="__RouteInfoPane" style="display:
                </td>
                <td>
                  <div id="__Map" style="position:relative;
                </td>
                <td>
                  <div id="__SearchTxtResultPane" style="ov
                </td>
              </tr>
            </table>
          </td>
        </tr >
      </table>
    </div>
  </ekoutput>
</ekmarkup>
```

When using the map.ekml file remember these 3 rules.

1. Do not change any of the IDs.
2. You can change any tag's style information.
3. You can move the variables in the third table row around to create different layouts. For example, you can display these sections horizontally or vertically.

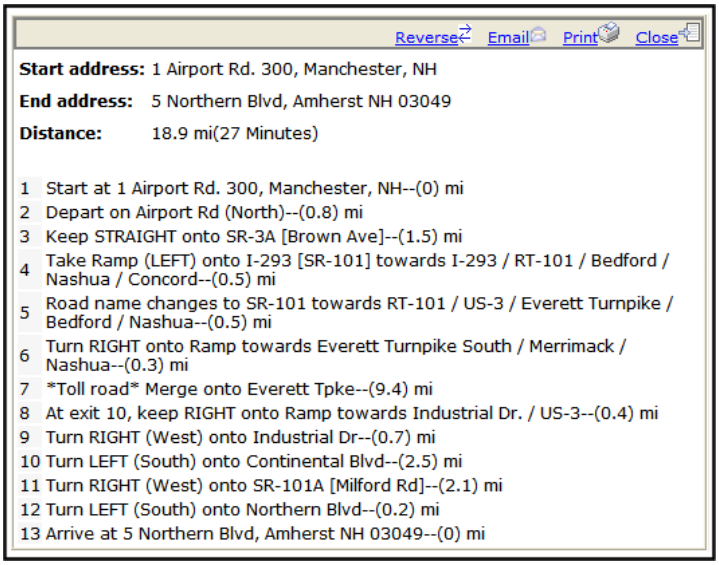
__Map

Displays the map section of the Map server control.



__RouteInfoPane

The panel that displays driving directions from the starting address to the arrival address.



Reverse Email Print Close

Start address: 1 Airport Rd. 300, Manchester, NH

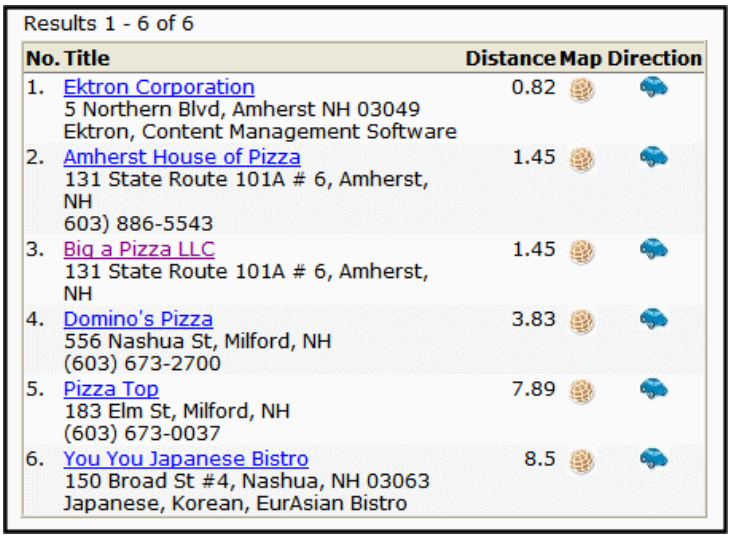
End address: 5 Northern Blvd, Amherst NH 03049

Distance: 18.9 mi(27 Minutes)

1 Start at 1 Airport Rd. 300, Manchester, NH--(0) mi
2 Depart on Airport Rd (North)--(0.8) mi
3 Keep STRAIGHT onto SR-3A [Brown Ave]--(1.5) mi
4 Take Ramp (LEFT) onto I-293 [SR-101] towards I-293 / RT-101 / Bedford / Nashua / Concord--(0.5) mi
5 Road name changes to SR-101 towards RT-101 / US-3 / Everett Turnpike / Bedford / Nashua--(0.5) mi
6 Turn RIGHT onto Ramp towards Everett Turnpike South / Merrimack / Nashua--(0.3) mi
7 *Toll road* Merge onto Everett Tpke--(9.4) mi
8 At exit 10, keep RIGHT onto Ramp towards Industrial Dr. / US-3--(0.4) mi
9 Turn RIGHT (West) onto Industrial Dr--(0.7) mi
10 Turn LEFT (South) onto Continental Blvd--(2.5) mi
11 Turn RIGHT (West) onto SR-101A [Milford Rd]--(2.1) mi
12 Turn LEFT (South) onto Northern Blvd--(0.2) mi
13 Arrive at 5 Northern Blvd, Amherst NH 03049--(0) mi

Search Txt result pane

Displays the results pane from the search.



Results 1 - 6 of 6

No.	Title	Distance	Map	Direction
1.	Ektron Corporation 5 Northern Blvd, Amherst NH 03049 Ektron, Content Management Software	0.82		
2.	Amherst House of Pizza 131 State Route 101A # 6, Amherst, NH 603) 886-5543	1.45		
3.	Big a Pizza LLC 131 State Route 101A # 6, Amherst, NH	1.45		
4.	Domino's Pizza 556 Nashua St, Milford, NH (603) 673-2700	3.83		
5.	Pizza Top 183 Elm St, Milford, NH (603) 673-0037	7.89		
6.	You You Japanese Bistro 150 Broad St #4, Nashua, NH 03063 Japanese, Korean, EurAsian Bistro	8.5		

messageboard.ekml

This file defines which items and information are included when displaying message information for a message board using the *MessageBoard* on page 2270 server control.

messageboard.ekml variables

- *[$AddCommentBox]* on page 2654. The Add Comment text box and button for a message board.
- *[$ApproveMessageLink]* on page 2655. An Approve link to approve comments when message board moderation is active.
- *[$Avatar]* on page 2655. The profile image of the member who entered the comments on the message board.
- *[$DateCreated]* on page 2659. The date the content was created.
- *[$DateModified]* on page 2660. The date the content was last modified.
- *[$DeleteMessageLink]* on page 2660. The Delete link for a comment on the message board. Only Administrators, the person who left the message or the person who owns the board.
- *[$DisplayName]* on page 2661. The display name of the member who left the message.
- *[$EmailAddress]* on page 2663. The email address of the member who left the message on the board.
- *[$FirstName]* on page 2664. The first name of the person who left the comment on the message board.
- *[$LastName]* on page 2669. The last name of the person who left the comment on the message board.
- *[$MessageText]* on page 2670. The message on a message board.
- *[$NumberComments]* on page 2670. The number of comments posted to a message board.

- *[$UserName]* on page 2681. The username of a user who left a comment on a message board.

metadatalist.ekml

This file defines which items and information are included when displaying content item information for a metadata list using the *MetadataList* on page 2503 server control.

metadatlist.ekml variables

- *[$Comment]* on page 2658. The content's comment information.
- *[$ContentId]* on page 2659. The content item's ID.
- *[$DateCreated]* on page 2659. The date the content was created.
- *[$DateModified]* on page 2660. The date the content was last modified.
- *[$EditorFirstName]* on page 2662. The last editor's first name for a content item.
- *[$EditorLastName]* on page 2662. The last editor's last name for a content item.
- *[$FolderId]* on page 2665. The folder ID of a content item.
- *[$Html]* on page 2666. The HTML contained in the content item.
- *[$HyperLink]* on page 2666. Adds a hyperlink using the title of the content block as the text.
- *[$Image]* on page 2667. The path for the image defined in a content item's Metadata. When wrapped in `<img src=""/>` tag, the image appears.
- *[$ImageIcon]* on page 2667. An image icon for the content item. For example, if the content item is HTML, the () icon appears.
- *[$ImageThumbnail]* on page 2667. The path for the image's thumbnail defined in a content item's Metadata.
- *[$Index]* on page 2668. Serialize the content items in a numbered list.
- *[$ItemCount]* on page 2668. Total number of items in a list.
- *[$Language]* on page 2668. Language ID for the content item.
- *[$LinkTarget]* on page 2669. When added to an `<a href="">` tag's `target=""` attribute, this variable reads the server control's `LinkTarget` property and uses its setting.
- *[$QuickLink]* on page 2672. Quicklink information for the content item. When wrapped in an `<a href="">` tag, you can create a Hyperlink.
- *[$SERVER_NAME]* on page 2675. The server name. For example, If this variable is applied to `http://www.example.com/demo.aspx`, the return is `www.example.com`.
- *[$ShowBubble(width,height)]* on page 2677. Similar to `[$ShowBubble]`. It calls the `<ekbubbleinfo>` tags and lets you set the width of the bubble.
- *[$ShowBubble]* on page 2676. Calls the `<ekbubbleinfo>` tags and places the information contained within those tags in a pop-up bubble.

- *[$ShowContent('htmltagid')]* on page 2677. Calls the <ekcontentinfo> tags an places the information in those tags within the specified HTML tag ID. Replace the 'htmltagid' with the ID of the tag.
- *[$Status]* on page 2678. The status of a content item.
- *[$Teaser]* on page 2678. The content item's summary information. If the item is an HTML form, this variable is *not* supported with this ekml file.
- *[$Title]* on page 2679. The content item's title.
- *[$UrlEncode('str')]* on page 2679. Encodes the string information. This variable can be used to encode another EkML variable and place it in an email.
- *[$UrlParam('paramname')]* on page 2680. The value of a QueryString's parameter. For example, if the QueryString is `?id=27` and the variable is `[$UrlParam('id')]`, 27 appears.

taxonomy.ekml

This file defines which items and information are included when displaying taxonomy item information using the *Directory* on page 2348 server control. Similar to `maps.ekml`, `taxonomy.ekml` works differently than other .ekml files.

The Directory server control produces multiple areas where content or functionality is defined. To specify these areas in a template, you would call the `<ekoutput>` tag with the mode attribute equaling the area you want to define. For example, in the Directory server control, you can define information in the breadcrumb area of the server control by using `<ekoutput mode="breadcrumb">`.

Additional descriptions and commenting in the `taxonomy.ekml` file will assist you with learning about EkML with taxonomy.

Taxonomy <ekoutput> modes

The following <ekoutput modes=""> are used with the `taxonomy.ekml` template.

- `<ekoutput mode="breadcrumb">`. Defines the display of the breadcrumb portion of the taxonomy. Within these tags you can define:
  - `<bctitle>`. Title for the breadcrumbs
  - `<bcrootlink>`. Root link text
  - `<bcseparator>`. Separator used to between breadcrumbs in the breadcrumb trail
  - `<bchyperlink>`. Hyperlinks in the breadcrumb trail
  - `<bcactivelink>`. The current active breadcrumb item
- `<ekoutput mode="category">`. Defines how the category information appears. Within these tags you can define:
- `<ekcolrepeat>` takes the category links and spreads them over the amount of columns that are defined in the Directory server control's `TaxonomyCol` property. This tag must appear within `<ekrepeat>` tags.
- `<ekoutput mode="categorybacklink">`. Defines information about the link that moves the category up one level. This can be a text link or an image link. In this mode, you need to define the following tags:

- <ekactivebacklink> </ekactivebacklink>. Between these tags, define what appears when a user is in a sub category. Add the [$categorybacklink] variable between these tags and an clickable image appears that lets a user navigate one level up.
- <ekdisablebacklink> </ekdisablebacklink>. Between these tags, define what happens when a user is at the top level category. Add the [$categorybacklink] variable between these tags with an <a> tag have users navigate to another location. For example:

```
<a href="http://www.example.com" target="_blank">
[$categorybacklink]</a>
```

See also: *[$CategoryBackLink]* on page 2656

- `<ekoutput mode="article_search">`. Defines how the search results from the taxonomy are displayed.
- `<ekoutput mode="article">`. Defines the display of an individual content item in your taxonomy.
- `<ekoutput mode="view">`. This section defines the overall output view of the Taxonomy. It denotes the locations of the Search box, Breadcrumb, Category and Articles.

taxonomy.ekml variables

- *[$AddArticle]* on page 2653. Adds a link that allows a logged in user to add HTML content to Ektron.
- *[$AddAsset]* on page 2653. Adds a link that allows a logged in user to add assets to Ektron.
- *[$CategoryBackLink]* on page 2656. When placed between `<ekactivebacklink>` tags, this variable adds a clickable image that allows a user to navigate up one category level. When placed between `<ekdisablebacklink>` tags and in the text area of an `<a>` tag, this variable adds a clickable image that allows a user to navigate to another URL once they reach the top level category.
- [$CategoryID]. The ID of a taxonomy's category.
- *[$ContentByteSize]* on page 2659. The content item's size in KB. Works only with taxonomy search. `<ekoutput mode="article_search">`
- *[$ContentId]* on page 2659. The content item's ID.
- *[$DateCreated]* on page 2659. The date the content was created.
- *[$DateModified]* on page 2660. The date the content was last modified.
- *[$EditorFirstName]* on page 2662. The last editor's first name for a content item.

- *[$EditorLastName]* on page 2662. The last editor's last name for a content item.
- *[$FolderId]* on page 2665. The folder ID of a content item.
- *[$Html]* on page 2666. The HTML contained in the content item.
- *[$HyperLink]* on page 2666. Adds a hyperlink using the title of the content block as the text.
- *[$ImageIcon]* on page 2667. An image icon for the content item type. For example, if the content item is HTML, the () icon appears.
- *[$Index]* on page 2668. Serialize the content items in a numbered list.
- *[$ItemCount]* on page 2668. The total number of items in a list.
- *[$Language]* on page 2668. The language ID for the content item.
- *[$LinkTarget]* on page 2669. When added to an `<a href="">` tag's `target=""` attribute, this variable reads the server control's `LinkTarget` property and uses its setting.
- *[$PagingCurrentEndIndex]* on page 2671. The end count number of the items on the page. For example, if you are displaying items 11-20 on a page, this variable represents the number 20. Works only with taxonomy search. `<ekoutput mode="article_search">`
- *[$PagingCurrentStartIndex]* on page 2672. The numerical record of the first item on a page. For example, if you are displaying items 1 through 10 out of 50 total items on a page, this variable represents the number 1. Works only with taxonomy search. `<ekoutput mode="article_search">`
- *[$QuickLink]* on page 2672. The Quicklink information for the content item. When wrapped in an `<a href="">` tag, you can create a Hyperlink.
- *[$SearchDuration]* on page 2673. The amount of time, in seconds, it has taken to execute the search. Works only with taxonomy search. `<ekoutput mode="article_search">`
- *[$SearchSummary]* on page 2674. Creates a summary from information stored in the indexing service for each item in the search results. Works only with taxonomy search. `<ekoutput mode="article_search">`
- *[$SearchText]* on page 2674. The text for which a user is searching. This information is same as what a user entered in the search text box. Works only with taxonomy search. `<ekoutput mode="article_search">`
- *[$SERVER_NAME]* on page 2675. The server name. For example, If this variable is applied to `http://www.example.com/demo.aspx`, the return is `www.example.com`.
- *[$ShowAllcategory]* on page 2675. Adds to the Taxonomy search screen a checkbox that lets the user decide whether to display categories that have no items.
- *[$ShowBubble]* on page 2676. Calls the `<ekbubbleinfo>` tags and places the information contained within those tags in a pop-up bubble.
- *[$ShowBubble(width,height)]* on page 2677. Similar to `[$ShowBubble]`. It calls the `<ekbubbleinfo>` tags and lets you set the width of the bubble.

- *[$ShowContent('htmltagid')]* on page 2677. Calls the <ekcontentinfo> tags an places the information in those tags within the specified HTML tag ID. Replace the 'htmltagid' with the ID of the tag.
- *[$Status]* on page 2678. The status of a content item.
- *[$Teaser]* on page 2678. The content item's summary information. If the item is an HTML form, this variable is *not* supported with this ekml file.
- *[$TemplateQuickLink]* on page 2679. The Template Quicklink information assigned to the taxonomy item in the Workarea. When wrapped in an `<a href="">` tag, you can create a Hyperlink.
- *[$Title]* on page 2679. The content item's title.
- *[$UrlEncode('str')]* on page 2679. Encodes the string information. This variable can be used to encode another EkML variable and place it in an email.
- *[$UrlParam('paramname')]* on page 2680. The value of a QueryString's parameter. For example, if the QueryString is `?id=27` and the variable is `[$UrlParam('id')]code, 27 appears.`

websearch.ekml

This file defines which items and information are included when displaying search results using the WebSearch server control.

IMPORTANT: Deprecated. The WebSearch server control is deprecated as of Release 8.50. Instead of the WebSearch control, you should convert to templated search. If you are already using the WebSearch server control, you can continue to do so with one important exception: the Solr search provider is incompatible with the WebSearch server control.

The `websearch.ekml` template needs 2 `<ekoutput>` nodes.

- The first `<ekoutput>` node formats the results of non-image searches
- The second `<ekoutput>` node formats results that include images

websearch.ekml variables

- *[$ContentByteSize]* on page 2659. The size of the content item.
- *[$ContentId]* on page 2659. The content ID number assigned to the content.
- *[$DateModified]* on page 2660. The date the content was last modified.
- *[$EditorFirstName]* on page 2662. The last editor's first name for a content item.
- *[$EditorLastName]* on page 2662. The last editor's last name for a content item.
- *[$Image]* on page 2667. The path for the image defined in a content item's Metadata. When wrapped in `<img src="" />` tag, the image appears.
- *[$ImageIcon]* on page 2667. Used in non-image searches to display an image icon for the content item. For example, if the content item is HTML, the () icon appears.
- *[$ImageThumbnail]* on page 2667. The path for the image's thumbnail defined in a content item's Metadata.
- *[$ItemCount]* on page 2668. The total number of results produced by the search.

- *[$LinkTarget]* on page 2669. When added to an `<a href="">` tag's `target=""` attribute, this variable reads the server control's `LinkTarget` property and uses its setting.
- *[$PagingCurrentEndIndex]* on page 2671. The end count number of the items on the page. For example, if you are displaying items 11-20 on a page, this variable represents the number 20.
- *[$PagingCurrentStartIndex]* on page 2672. The numerical record of the first item on a page. For example, if you are displaying items 1 through 10 out of 50 total items on a page, this variable represents the number 1.
- *[$QuickLink]* on page 2672. The Quicklink information for the content item. When wrapped in an `<a href="">` tag, you can create a Hyperlink.
- *[$SearchDuration]* on page 2673. The amount of time, in seconds, it has taken to execute the search.
- *[$SearchSummary]* on page 2674. Creates an abstract for each item in the search results.

  NOTE: The Adobe IFilter, which is used to generate the abstract, is only supported in Tier 1 languages (English, French, German, and Japanese). If your website uses other languages, the abstract may not be legible. In such a case, you should suppress the abstract from the search results.

- *[$SearchText]* on page 2674. The text for which a user is searching. This information is same as what a user entered in the search text box.
- *[$ShortDateModified]* on page 2675. The date the content was modified. To display the date and time a content item was modified, use *[$DateModified]* on page 2660.
- *[$Title]* on page 2679. The content item's title.

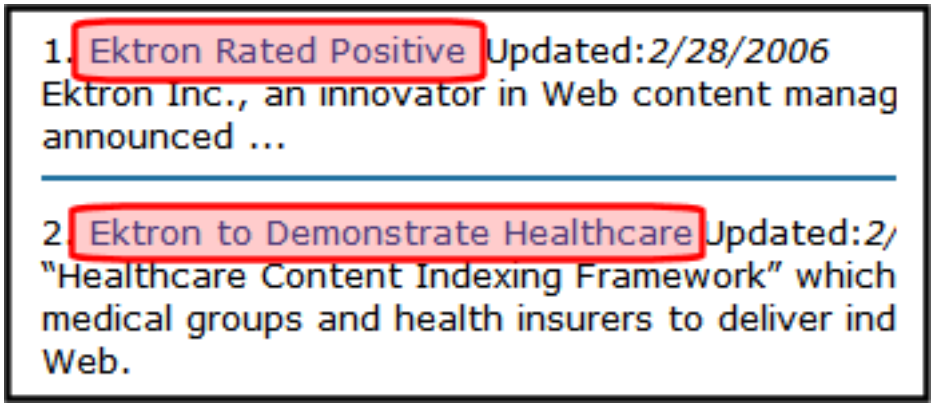
EkML example

By customizing the provided EkML templates, you can create a custom layout for the content you are displaying. For example, if you have a collection and you want to display it as a numbered list containing a content's hyperlink, the date it was last updated and its summary, you would create the following .ekml file and assign that file to a Collection server control's `MarkupLanguage` property.

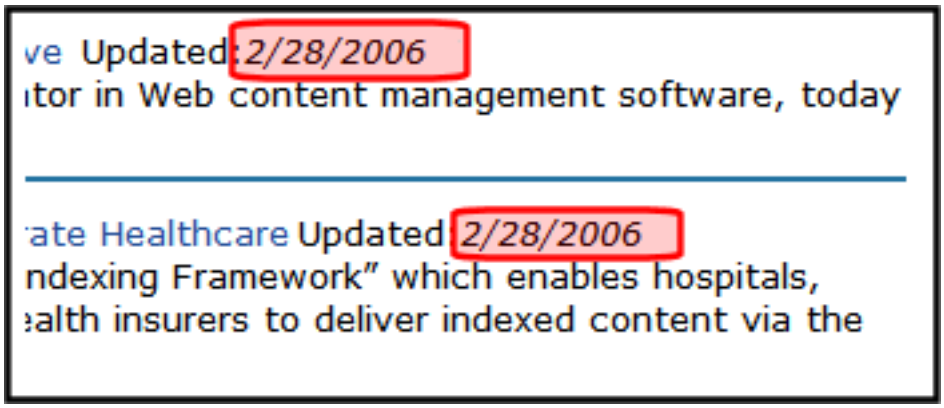
```
<ekmarkup>
  <ekoutput>
  <table width="100%"  border="0">
  <ekrepeat>
  <tr>
  <td> [$Index]. [$HyperLink] Updated:<i>[$DateModified]</i><br/>[$Teaser]
<hr/>
  </td>
  </tr>
  </ekrepeat>
  </table>
  </ekoutput>
</ekmarkup>
```

- The `<ekmarkup>` tags open and close the markup language. Everything to do with the EkML needs to be between these tags.
- The specific information you want displayed and any HTML formatting are added between the `<ekoutput>` tags.
- `<table width="100%" border="0"></table>` sets up a table. This is HTML formatting.
- The `<ekrepeat>` tags contain formatting information and variables for items in the Collection. It repeats this information for each item in the list.
- The `[$Index]` variable creates a numbered list for each content item in the Collection. Note, you can add a period (.) or other separator depending on how you want the list to look.

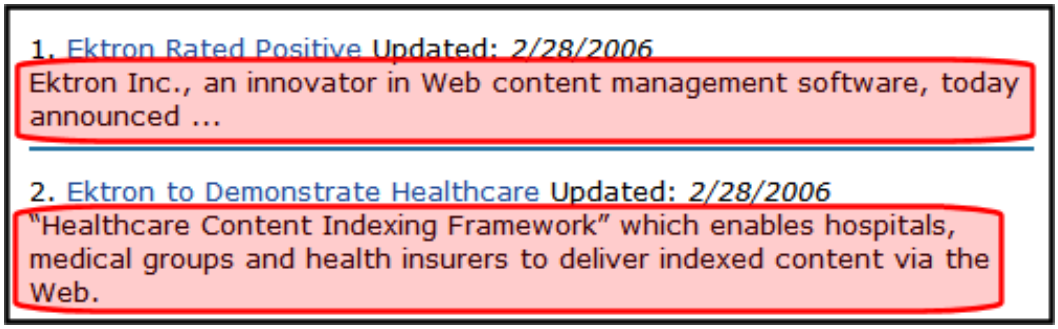
- The `[$HyperLink]` variable adds a hyperlink for each content item in the collection. The hyperlink use the content's title as the text for the link. There is no need for anchor tags, the variable creates them for you. If you want to use anchor tags for formatting you own links, you can use the `[$QuickLink]` variable.



- `Updated:` is plain text.
- The `<i></i>` tags are HTML that cause the `[$DateModified]` variable to appear in italics.
- The `[$DateModified]` variable displays the date and time each content item in the Collection was last modified.

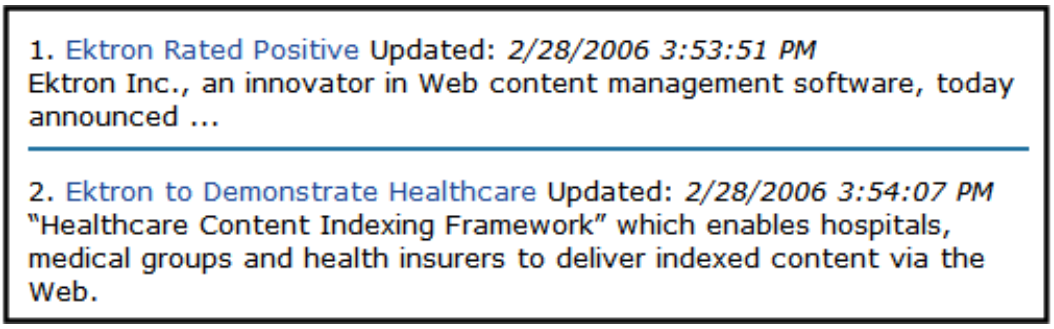


- `<br/>` is HTML. It adds a line break.
- The `[$Teaser]` variable displays the summary for each content item in the Collection.



- `<hr/>` is HTML. It adds a horizontal rule line.

When the Collection appears, it is formatted as follows.



Because you can use HTML in the Ektron Markup Language, you can format the variables using common HTML tags.

EkML tags

The EkML tags define functions that occur when using the language to display content.

- **<ekmarkup>**. Use these tags to open and close the Ektron Markup Language.
- **<ekoutput>**. Define what information is output from the server control between these tags.

- **<ekrepeat>**. These tags cause what ever formatting information that appears between them to be applied to each item in the list. They always appear between the `<ekoutput>` tags.
- **<ekbubbleinfo>**. Creates a pop-up bubble. This tag is invoked when you use the `[$ShowBubble]` or `[$ShowBubble(width)]` variables. By adding different variables between the `<ekbubbleinfo>` tags, you define what information appears in the bubble. These tags are placed outside the `<ekoutput>` tags but within the `<ekmarkup>` tags.
- **<ekcontentinfo>**. Places the information defined between the tags in the specified HTML tag ID. This tag is invoked when you use the `[$ShowContent('htmltagid')]` variable.

EkML variables

The Ektron Markup Language uses variables that appear between the tags in a template file to define the information that appears in a control's display. Some variable are used in more than one template. These are known as common EkML variables.

Another type of variable is a server control specific variable, these variables can only be used with a specific server control. For example, `[$SearchSummary]` can only be used in WebSearch.

EkML variable list

The following list shows EkML variables and whether the variable is common or specific.

Variable	Description	Common or Control Specific
[$AddCommentBox] on page 2654	Displays the Add Comment text box and button for a message board.	MessageBoard
[$AddArticle] on page 2653	Adds a link that allows a logged in user to add HTML content to Ektron.	Directory
[$AddAsset] on page 2653	Adds a link that allows a logged in user to add assets to Ektron.	Directory
[$ApproveMessageLink] on page 2655	Displays an Approve link to approve comments when message board moderation is active.	MessageBoard
[$Avatar] on page 2655	Displays the profile image of the member who entered comments on a message board.	MessageBoard

Variable	Description	Common or Control Specific
[$CollectionDescription] on page 2657	Displays the collection's description.	Collection
[$CollectionTitle] on page 2657	Displays the collection's title.	Collection
[$Comment] on page 2658	The content's comment information appears.	Common Except: Map, WebSearch and taxonomy
[$ContentByteSize] on page 2659	Displays the content item's size in KB. Works only with taxonomy search. `<ekoutput mode="article_search">`. More information	Directory WebSearch
[$ContentId] on page 2659	Displays the content item's ID.	Common Except: Map
[$DateCreated] on page 2659	Displays the date the content was created.	Common Except: Map and WebSearch
[$DateModified] on page 2660	Displays the date and time the content was last modified.	Common Except: Map
[$DeleteMessageLink] on page 2660	Displays the Delete link for the comment on a message board.	MessageBoard
[$DisplayName] on page 2661	Displays the display name of the member who left the message on the board.	MessageBoard
[$EditorFirstName] on page 2662	Displays the last editor's first name for a content item.	Common Except: Map
[$EditorLastName] on page 2662	Displays the last editor's last name for a content item. 1	Common Except: Map
[$EmailAddress] on page 2663	Displays the email address of the member.	MessageBoard
[$FirstName] on page 2664	Displays the first name of the person who left the comment on a message board.	MessageBoard

Variable	Description	Common or Control Specific
[$FolderDescription] on page 2664	Displays the folder's description.	ListSummary
[$FolderId] on page 2665	Displays the folder ID of a content item.	Common Except: Map and WebSearch
[$FolderName] on page 2665	Displays the folder's name.	ListSummary
[$Html] on page 2666	Displays the HTML contained in the content item. "	Common Except: Map and WebSearch
[$HyperLink] on page 2666	Adds a hyperlink using the title of the content block as the text. More information:	Common Except: Map and WebSearch
[$Image] on page 2667	Displays the path for the image defined in a content item's Metadata. When wrapped in `<img src=""/>` tag, the image appears. For example: `<img src="[$Image]"/>`.	Common Except: Map
[$ImageIcon] on page 2667	Displays an image icon for the content item. For example, if the content item is HTML, the () icon appears.	Common Except: Map
[$ImageThumbnail] on page 2667	Displays the path for the image's thumbnail defined in a content item's Metadata. When wrapperd in `<img src=""/>` tag, a thumbnail version of the image appears. For example: `<img src="[$ImageThumbnail]"/>`.	Common Except: Map
[$Index] on page 2668	Serialize the content items in a numbered list.	Common Except: Map and WebSearch
[$ItemCount] on page 2668	The total number of items in a list.	Common Except: Map

Variable	Description	Common or Control Specific
[$Language] on page 2668	Displays the language ID for the content item.	Common Except: Map and WebSearch
[$LastName] on page 2669	Displays the last name of the person who left the comment on a message board.]	MessageBoard
[$LinkTarget] on page 2669	When added to an `<a href="">` tag's `target=""` attribute, this variable reads the server control's `LinkTarget` property and uses its setting.	Common Except: Map
[$MessageText] on page 2670	Displays the text of a message on a message board.	MessageBoard
[$NumberComments] on page 2670	Displays the number of comments posted to a message board.	MessageBoard
[$PagingCurrentEndIndex] on page 2671	The end count number of the items on the page. For example, if you are displaying items 11-20 on a page, this variable represents the number 20. Works only with taxonomy search. `<ekoutput mode="article_search">`.	Directory and WebSearch
[$PagingCurrentStartIndex] on page 2672	The numerical record of the first item on a page. For example, if you are displaying items 1 through 10 out of 50 total items on a page, this variable represents the number 1. Works only with taxonomy search. `<ekoutput mode="article_search">`.	Directory and WebSearch
[$QuickLink] on page 2672	Displays the Quicklink information for the content item. When wrapped in an `<a href="">` tag, you can create a Hyperlink.	Common Except: Map
[$SearchDuration] on page 2673	Displays the amount of time, in seconds, it has taken to execute the search. Works only with taxonomy search. `<ekoutput mode="article_search">`. More information:	Directory and WebSearch

Variable	Description	Common or Control Specific
[$SearchSummary] on page 2674	Creates a summary for each item in the search results. Works only with taxonomy search. `<ekoutput mode="article_search">`.	Directory and WebSearch
[$SearchText] on page 2674	Displays the text for which a user is searching. This information is same as what a user entered in the search text box. Works only with taxonomy search. `<ekoutput mode="article_search">`.	Directory and WebSearch
[$SERVER_NAME] on page 2675	Displays the server name. For example, If this variable is applied to `http://www.example.com/demo.aspx`, the return is `www.example.com`.	Common Except: Map and WebSearch
[$ShortDateModified] on page 2675	Displays the date the content was modified. To display the date and time a content item was updated, use *[$DateModified]* on page 2660	WebSearch
[$ShowAllcategory] on page 2675	Adds to the Taxonomy search screen a checkbox that lets the user decide whether to display categories that have no items.	Directory
[$ShowBubble(width,height)] on page 2677	This is similar to `[$ShowBubble]`. It calls the `<ekbubbleinfo>` tags and lets you set the width and height of the bubble. For example, [$ShowBubble(300,400)]. The first number represents the width. The second number represents the height.	Common Except: Map and WebSearch
[$ShowBubble] on page 2676	Calls the `<ekbubbleinfo>` tags and places the information contained within those tags in a pop-up bubble.	Common Except: Map and WebSearch
[$ShowContent('htmltagid')] on page 2677	Calls the `<ekcontentinfo>` tags an places the information in those tags within the specified HTML tag ID. Replace the htmltagid with the ID of the tag.	Common Except: Map and WebSearch
[$Status] on page 2678	Displays the status of a content item.	Common Except: Map and WebSearch

Variable	Description	Common or Control Specific
[$Teaser] on page 2678	Displays the content item's summary information. **NOTE:** If the item is an HTML form, this variable is *not* supported with this ekml file.	Common Except: Map and WebSearch
[$TemplateQuickLink] on page 2679	the Template Quicklink information assigned to the taxonomy item in the Workarea. When wrapped in an `<a href="">` tag, you can create a Hyperlink. See also: Creating a Taxonomy-based Search on a Web page. More information :	Directory
[$Title] on page 2679	Displays the content item's title.	Common Except: Map
[$UrlEncode('str')] on page 2679	Encodes the string information. Replace **str** with the string you want to encode. This variable can be used to encode another EkML variable and place it in an email.	Common Except: Map and WebSearch
[$UrlParam('paramname')] on page 2680	Displays the value of a query string's given parameter. For example, if the query string is `?id=27` and the variable is `[$UrlParam ('id')], 27 appears.`	Common Except: Map and WebSearch
[$UserName] on page 2681	Displays the Username of user who left a comment on a message board.	MessageBoard

[$AddArticle]

This variable adds a link that allows a user to add HTML content to Ektron. Clicking the link opens an editor. When the content is added, it's automatically added to the taxonomy category associated with the Directory server control. When you allow users to add content using this variable, you should set the `AddItemFolderID` property in the Directory server control to the folder ID where the content will be stored. This variable should not be added between the <ekrepeat></ekrepeat> tags.

```
<tr>
  <td>
    [$AddArticle]
  </td>
</tr>
```

[$AddAsset]

This variable adds a link that allows a user to add assets to Ektron via a drag and drop box. When the asset is added, it's automatically added to the taxonomy category associated with the Directory server control. When you allow users to add assets using this variable, you should set the `AddItemFolderID` property in the Directory server control to the folder ID where the asset will be stored. This variable should not be added between the <ekrepeat></ekrepeat> tags.

```
<tr>
  <td>
    [$AddAsset]
  </td>
</tr>
```

[$AddCommentBox]

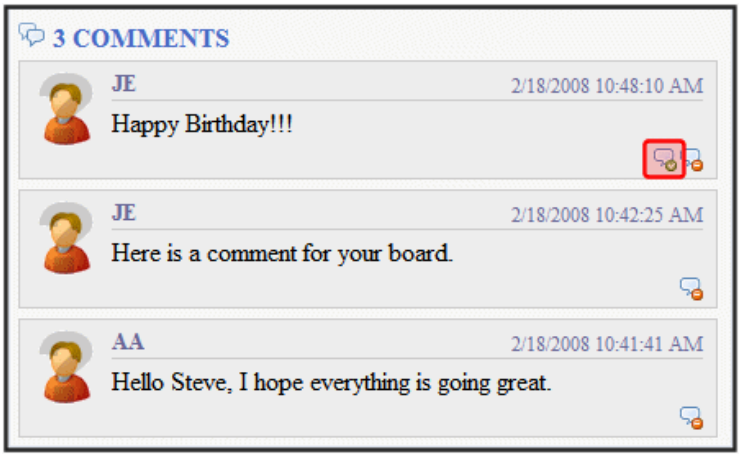
Displays the Add Comment text box and button for a message board.


```
<ekoutput>
  <div class="ContributionForm">
    <h4>[$NumberComments] Comments</h4>
    <ul>
      <ekrepeat>
        <li class="ekMessagePost">
          <div class="avatar">[$Avatar]</div>
          <div class="message">
            <div class="metaData">
              <span class="username">[$UserName]</span>
              <span class="time">[$DateCreated]</span>
            </div>
            <p>[$MessageText]</p>
            <ul class="commands">
              <li class="ekDeleteMessage">[$DeleteMessageLink]</li>
              <li class="ekApproveMessage">[$ApproveMessageLink]</li>
            </ul>
          </div>
        </li>
      </ekrepeat>
    </ul>
[$AddCommentBox]
```

```
  </div>
</ekoutput>
```

[$ApproveMessageLink]

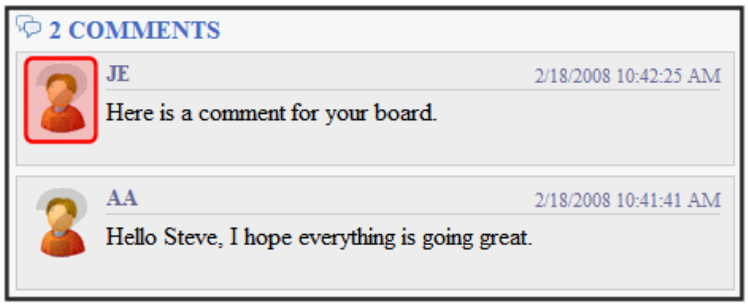
Displays the Approve link for the message. This link is used to approve the message for display when the `Moderate` property is set to true. Only Administrators, the person who left the message, or the person who owns the board can see this link.




```
<ekoutput>
  <div class="ContributionForm">
    <h4>[$NumberComments] Comments</h4>
    <ul>
      <ekrepeat>
        <li class="ekMessagePost">
          <div class="avatar">[$Avatar]</div>
          <div class="message">
            <div class="metaData">
              <span class="username">[$UserName]</span>
              <span class="time">[$DateCreated]</span>
            </div>
            <p>[$MessageText]</p>
            <ul class="commands">
              <li class="ekDeleteMessage">[$DeleteMessageLink]</li>
              <li class="ekApproveMessage">[$ApproveMessageLink]</li>
            </ul>
          </div>
        </li>
      </ekrepeat>
    </ul>
    [$AddCommentBox]
  </div>
</ekoutput>
```

[$Avatar]

Display the profile image of the member who entered the comments on the message board.




```
<ekoutput>
  <div class="ContributionForm">
    <h4>[$NumberComments] Comments</h4>
    <ul>
      <ekrepeat>
        <li class="ekMessagePost">
          <div class="avatar">[$Avatar]</div>
          <div class="message">
            <div class="metaData">
              <span class="username">[$UserName]</span>
              <span class="time">[$DateCreated]</span>
            </div>
            <p>[$MessageText]</p>
            <ul class="commands">
              <li class="ekDeleteMessage">[$DeleteMessageLink]</li>
              <li class="ekApproveMessage">[$ApproveMessageLink]</li>
            </ul>
          </div>
        </li>
      </ekrepeat>
    </ul>
    [$AddCommentBox]
  </div>
</ekoutput>
```

[$CategoryBackLink]

When placed between <ekactivebacklink> tags, this variable adds a clickable image that allows a user to navigate up one category level.

When placed between <ekdisablebacklink> tags and in the text area of an <a> tag, this variable adds a clickable image that lets a user navigate to another URL once they reach the top level category.

```
<ekoutput mode="categorybacklink">
  <ekactivebacklink>
    [$categorybacklink]
  </ekactivebacklink>
    <ekdisablebacklink>
    <a href="http://www.example.com" target="_blank">[$categorybacklink]</a>
    </ekdisablebacklink>
</ekoutput>
```

[$CollectionDescription]

Display the collection's description.


```
<ekmarkup>
<ekoutput>
<h3><b>[$CollectionTitle]</b></h3>
  <p/>[$CollectionDescription]<br/>
    <table width="100%"  border="0">
      <ekrepeat>
        <tr>
          <td>
            [$Index]. [$HyperLink]
          </td>
        </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$CollectionTitle]

Display the collection's title.


```
<ekmarkup>
  <ekoutput>
    <h3><b>[$CollectionTitle]</b></h3>
    <table width="100%"  border="0">
      <ekrepeat>
        <tr>
          <td>
            [$Index]. [$HyperLink]
          </td>
        </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

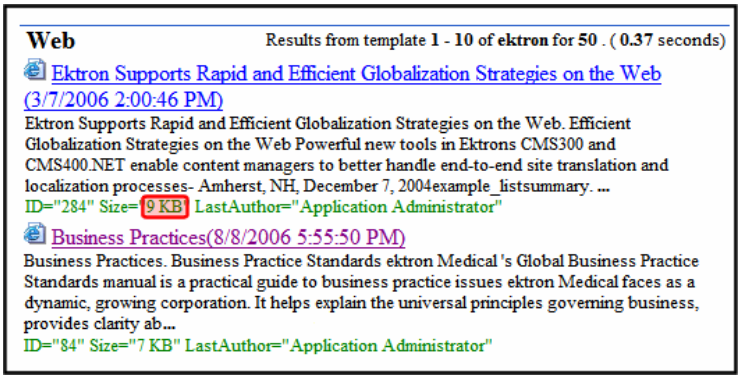
[$Comment]

The content's comment information appears. Comment information can be added in the Workarea via the content item's Comment tab.


```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$Index]. [$HyperLink] <br/> [$Comment]
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

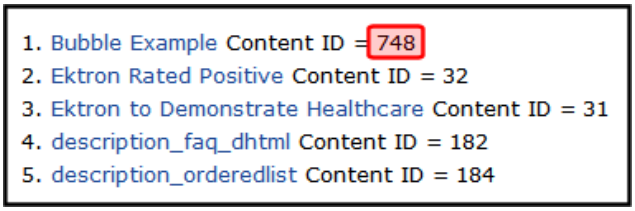
[$ContentByteSize]

Display the size of the content item in the results list.



[$ContentId]

Displays the content item's ID.




```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$Index]. [$HyperLink] Content ID = [$ContentId]
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

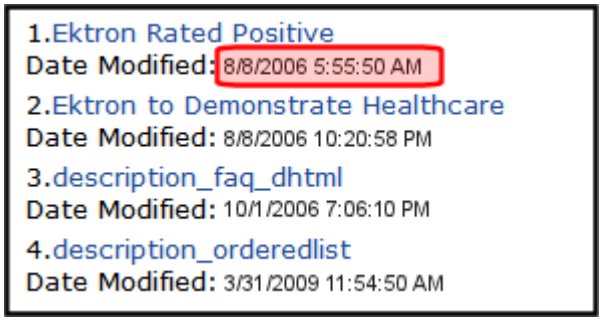
[$DateCreated]

Displays the date the content was created.

```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$Index]. [$HyperLink] - <i>[$DateCreated]</i>
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$DateModified]

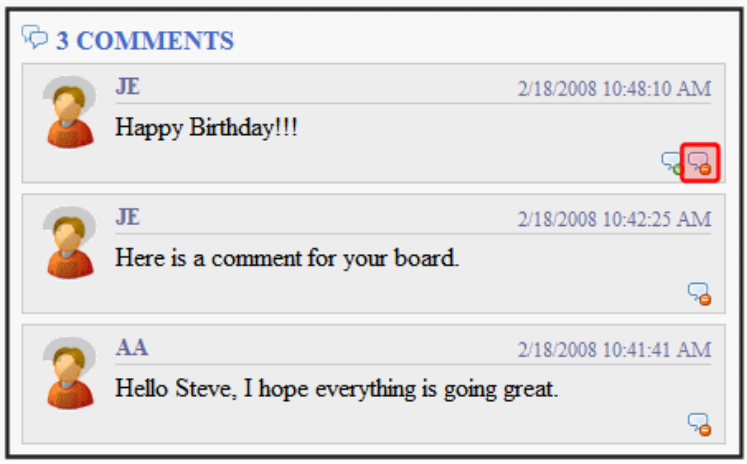
Displays the date and time the content was last modified. To display the date only, use *[$ShortDateModified]* on page 2675.




```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$Index]. [$HyperLink] <br/>Date Modified: <b>[$DateModified]</b>
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$DeleteMessageLink]

Displays the Delete link for a comment on the message board. Only Administrators, the person who left the message or the person who owns the board.




```
<ekoutput>
  <div class="ContributionForm">
    <h4>[$NumberComments] Comments</h4>
    <ul>
      <ekrepeat>
        <li class="ekMessagePost">
          <div class="avatar">[$Avatar]</div>
          <div class="message">
            <div class="metaData">
              <span class="username">[$UserName]</span>
              <span class="time">[$DateCreated]</span>
            </div>
            <p>[$MessageText]</p>
            <ul class="commands">
              <li class="ekDeleteMessage">[$DeleteMessageLink]</li>
              <li class="ekApproveMessage">[$ApproveMessageLink]</li>
            </ul>
          </div>
        </li>
      </ekrepeat>
    </ul>
    [$AddCommentBox]
  </div>
</ekoutput>
```

[$DisplayName]

Displays the display name of the member who left the message.

```
<ekoutput>
  <div class="ContributionForm">
    <h4>[$NumberComments] Comments</h4>
    <ul>
      <ekrepeat>
        <li class="ekMessagePost">
          <div class="avatar">[$Avatar]</div>
          <div class="message">
            <div class="metaData">
              <span class="username">[$DisplayName]</span>
              <span class="time">[$DateCreated]</span>
            </div>
            <p>[$MessageText]</p>
            <ul class="commands">
              <li class="ekDeleteMessage">[$DeleteMessageLink]</li>
              <li class="ekApproveMessage">[$ApproveMessageLink]</li>
            </ul>
          </div>
        </li>
      </ekrepeat>
    </ul>
    [$AddCommentBox]
  </div>
</ekoutput>
```

[$EditorFirstName]

Displays the last editor's first name for a content item.


```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$Index]. [$HyperLink] Last Editor: [$EditorFirstName]
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$EditorLastName]

Displays the last editor's last name for a content item.


```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$Index]. [$HyperLink] Last Editor: [$EditorLastName]
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$EmailAddress]

Displays the email address of the member who left the message on the board. To create a hyperlinked email, wrap the [$EmailAddress] variable in a <a> tag with the mailto: variable. For example:

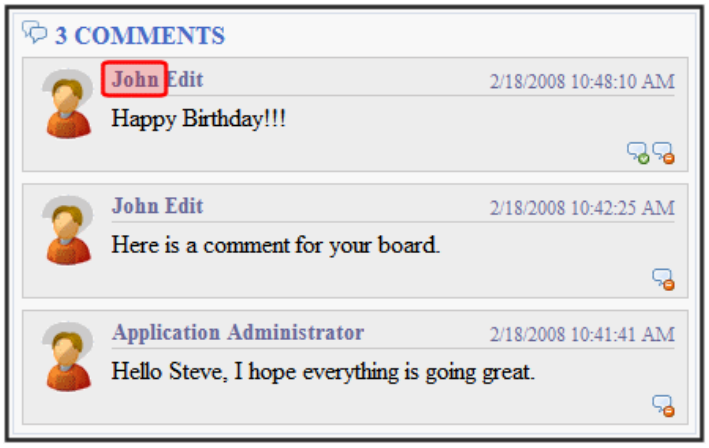
```
<a href="Mailto:[$EmailAddress]">[$EmailAddress]</a>
```


```
<ekoutput>
  <div class="ContributionForm">
    <h4>[$NumberComments] Comments</h4>
    <ul>
      <ekrepeat>
        <li class="ekMessagePost">
          <div class="avatar">[$Avatar]</div>
          <div class="message">
            <div class="metaData">
              <span class="username">[$UserName]  
              <a href="Mailto:[$EmailAddress]">[$EmailAddress]</a>
            </span>
              <span class="time">[$DateCreated]</span>
            </div>
            <p>[$MessageText]</p>
            <ul class="commands">
              <li class="ekDeleteMessage">[$DeleteMessageLink]</li>
```

```
            <li class="ekApproveMessage">[$ApproveMessageLink]</li>
          </ul>
        </div>
      </li>
    </ekrepeat>
  </ul>
  [$AddCommentBox]
 </div>
</ekoutput>
```

[$FirstName]

Displays the first name of the user who left a comment on a message board.




```
<ekoutput>
  <div class="ContributionForm">
    <h4>[$NumberComments] Comments</h4>
    <ul>
      <ekrepeat>
        <li class="ekMessagePost">
          <div class="avatar">[$Avatar]</div>
          <div class="message">
            <div class="metaData">
              <span class="username">[$FirstName] [$LastName]</span>
              <span class="time">[$DateCreated]</span>
            </div>
            <p>[$MessageText]</p>
            <ul class="commands">
              <li class="ekDeleteMessage">[$DeleteMessageLink]</li>
              <li class="ekApproveMessage">[$ApproveMessageLink]</li>
            </ul>
          </div>
        </li>
      </ekrepeat>
    </ul>
    [$AddCommentBox]
  </div>
 </ekoutput>
```

[$FolderDescription]

Displays the folder's description.

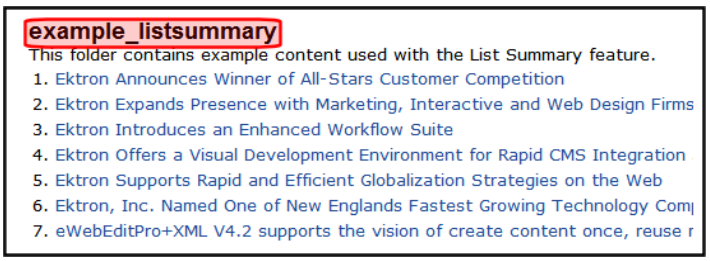

```
<ekmarkup>
  <ekoutput>
    <h3><b>[$FolderName]</b></h3><p/>[$FolderDescription]<br/>
    <table width="100%"  border="0">
      <ekrepeat>
        <tr>
          <td>
            [$Index]. [$HyperLink]
          </td>
        </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$FolderId]

Displays the folder ID of a content item.

[$FolderName]

Displays the folder's name.

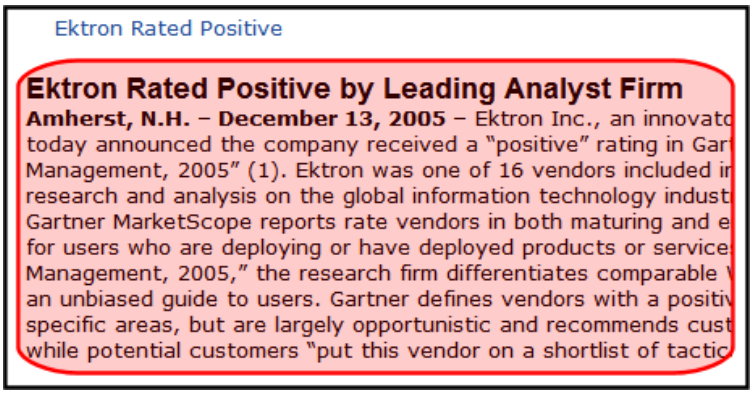



```
<ekmarkup>
  <ekoutput>
    <h3><b>[$FolderName]</b></h3><p/>[$FolderDescription]<br/>
    <table width="100%"  border="0">
      <ekrepeat>
        <tr>
          <td>
            [$Index]. [$HyperLink]
          </td>
        </tr>
      </ekrepeat>
    </table>
```

```
  </ekoutput>
</ekmarkup>
```

[$Html]

Displays the HTML contained in the content item. In the server control, the control's `GetHTML` property must be set to **True**; otherwise nothing appears. The exception to this is when the `[$Html]` appears between the `<ekbubbleinfo>` tags. In that case, the `GetHTML` property can be set to **True** or **False**.




```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$HyperLink]<br/>[$Html]
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$HyperLink]

Adds a hyperlink using the title of the content block as the text. You do not add an `<a href="">` tag when using this variable. That functionality is built into the EkML. Use this variable when you do not want to create a custom hyperlink. If you want to create a custom hyperlink, use the `[$QuickLink]` variable. See also: *[$QuickLink]* on page 2672


```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
```

```
      <ekrepeat>
      <tr>
        <td>
          [$Index]. [$HyperLink]
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$Image]

Displays the path for the image defined in a content item's Metadata. When wrapped in `<img src=""/>` tag, the image appears. For example, `<img src="[$Image]"/>`.

[$ImageIcon]

Displays an image icon for the content item. Except for HTML content, these icons are the same icons used in the Workarea to show the content type. HTML content uses the Internet Explorer icon (). For example, Forms use the Form icon ().

When using this variable with the server control, the `IncludeIcons` property is automatically set to **True**.

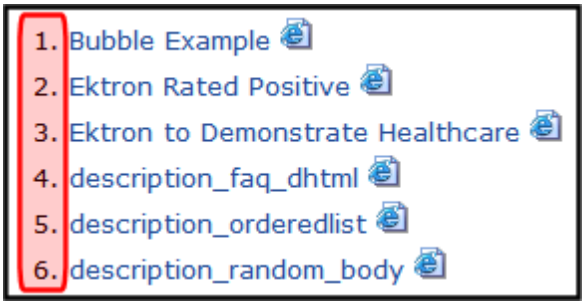

```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$Index]. [$HyperLink] [$ImageIcon]
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$ImageThumbnail]

Displays the path for the image's thumbnail defined in a content item's Metadata. When wrapped in `<img src=""/>` tag, the image thumbnail appears. For example, `<img src="[$ImageThumbnail]"/>`.

[$Index]

Serialize the content items in a numbered list.



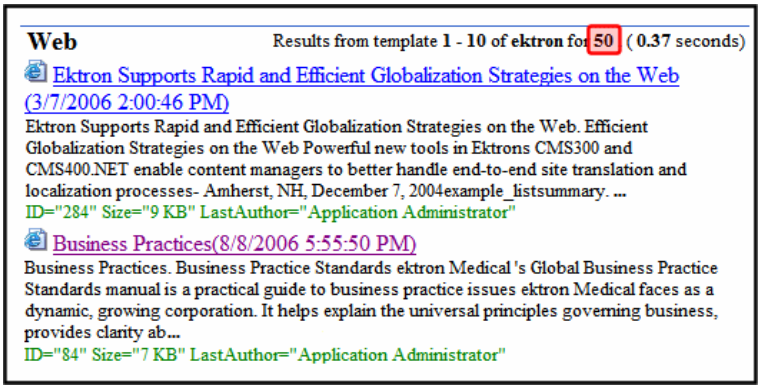

```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$Index]. [$HyperLink] [$ImageIcon]
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$ItemCount]

Displays the total number of results in a list that are produced by the search. In the server control, the `EnablePaging` property must be set to **True**; otherwise nothing appears.

This variable is typically used in the following context:

Results `[$PagingCurrentStartIndex]`.`[$PagingCurrentEndIndex]` of **`[$ItemCount]`** for item `[$SearchText] ([$SearchDuration])`.



[$Language]

Displays the language ID for the content item.

[$LastName]

Displays the last name of the user who left a comment on a message board.


```
<ekoutput>
    <div class="ContributionForm">
      <h4>[$NumberComments] Comments</h4>
      <ul>
        <ekrepeat>
          <li class="ekMessagePost">
            <div class="avatar">[$Avatar]</div>
            <div class="message">
              <div class="metaData">
                <span class="username">[$FirstName] [$LastName]</span>
                <span class="time">[$DateCreated]</span>
              </div>
              <p>[$MessageText]</p>
              <ul class="commands">
                <li class="ekDeleteMessage">[$DeleteMessageLink]</li>
                <li class="ekApproveMessage">[$ApproveMessageLink]</li>
              </ul>
            </div>
          </li>
        </ekrepeat>
      </ul>
      [$AddCommentBox]
    </div>
  </ekoutput>
```

[$LinkTarget]

When added to an `<a href="">` tag's `target=""` attribute, this variable reads the server control's `LinkTarget` property and uses its setting. For example, If you want to create a custom hyperlink that opens in a new window, you set the server control's `LinkTarget` property to `_Blank`. Then, in the Ektron Markup Language file, add the `[$LinkTarget]` variable to the `<a href="">` tag's `Taget=""` attribute. A code example appears below.

```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
```

```
      <ekrepeat>
      <tr>
        <td>
         <a href="[$QuickLink]" target="[$LinkTarget]">[$Title]</a>
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$MessageText]

Displays the text of a message on a message board.


```
<ekoutput>
    <div class="ContributionForm">
      <h4>[$NumberComments] Comments</h4>
      <ul>
        <ekrepeat>
          <li class="ekMessagePost">
            <div class="avatar">[$Avatar]</div>
            <div class="message">
              <div class="metaData">
                <span class="username">[$UserName]</span>
                <span class="time">[$DateCreated]</span>
              </div>
              <p>[$MessageText]</p>
              <ul class="commands">
                <li class="ekDeleteMessage">[$DeleteMessageLink]</li>
                <li class="ekApproveMessage">[$ApproveMessageLink]</li>
              </ul>
            </div>
          </li>
        </ekrepeat>
      </ul>
      [$AddCommentBox]
    </div>
  </ekoutput>
```

[$NumberComments]

Displays the number of comments posted on the message board.

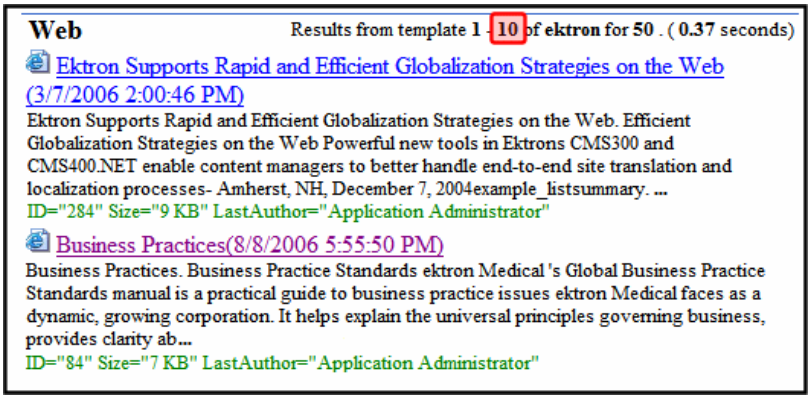

```
<ekoutput>
  <div class="ContributionForm">
    <h4>[$NumberComments] Comments</h4>
    <ul>
      <ekrepeat>
        <li class="ekMessagePost">
          <div class="avatar">[$Avatar]</div>
          <div class="message">
            <div class="metaData">
              <span class="username">[$UserName]</span>
              <span class="time">[$DateCreated]</span>
            </div>
            <p>[$MessageText]</p>
            <ul class="commands">
              <li class="ekDeleteMessage">[$DeleteMessageLink]</li>
              <li class="ekApproveMessage">[$ApproveMessageLink]</li>
            </ul>
          </div>
        </li>
      </ekrepeat>
    </ul>
    [$AddCommentBox]
  </div>
</ekoutput>
```

[$PagingCurrentEndIndex]

The numerical record of the last item on a page. For example, if you are displaying items 1 through 10 out of 50 total items on a page, this variable represents the number 10.

This variable is typically used in the following context:

Results `[$PagingCurrentStartIndex]`. **`[$PagingCurrentEndIndex]`** of `[$ItemCount]` for item `[$SearchText] ([$SearchDuration])`.

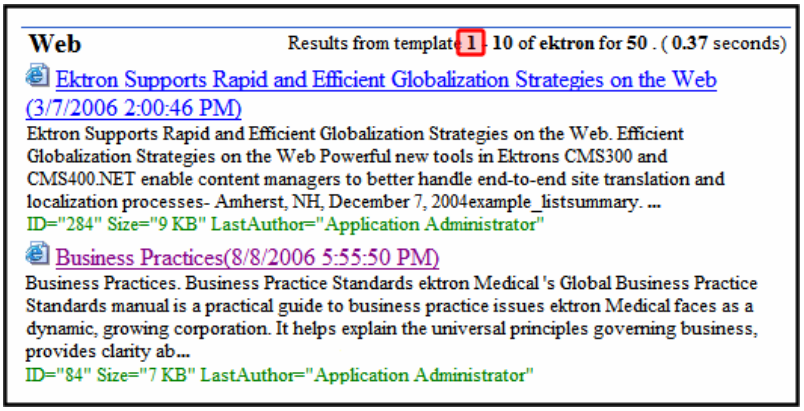


[$PagingCurrentStartIndex]

The numerical record of the first item on a page. For example, if you are displaying items 1 through 10 out of 50 total items on a page, this variable represents the number 1.

This variable is typically used in the following context:

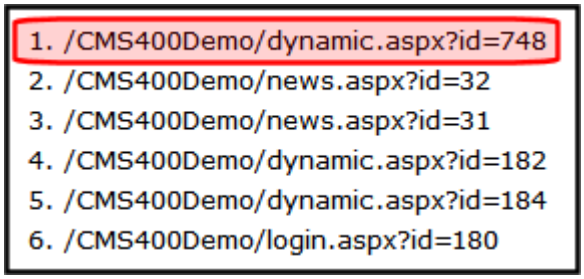
Results **[$PagingCurrentStartIndex]**. `[$PagingCurrentEndIndex]` of `[$ItemCount]` for item `[$SearchText] ([$SearchDuration])`.



[$QuickLink]

Displays the Quicklink information for the content item. When wrapped in an `<a href=””>` tag, you can create a Hyperlink. Use this property instead of the `[$HyperLink]` if you want to customize your hyperlinks. See also: *[$HyperLink]* on page 2666

The first image shows the variable displaying the Quicklink information. The second image shows the Quicklink as a Hyperlink with a custom text.




```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$Index]. [$QuickLink]
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

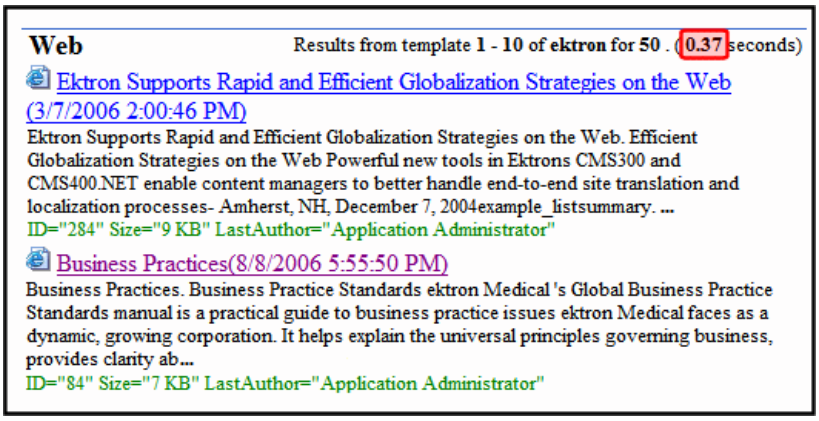

```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$Index]. <a href="[$QuickLink]">Click Here For Item #[$Index]</a>
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$SearchDuration]

Displays the amount of time, in seconds, it has taken to perform the search.

This variable is typically used in the following context:

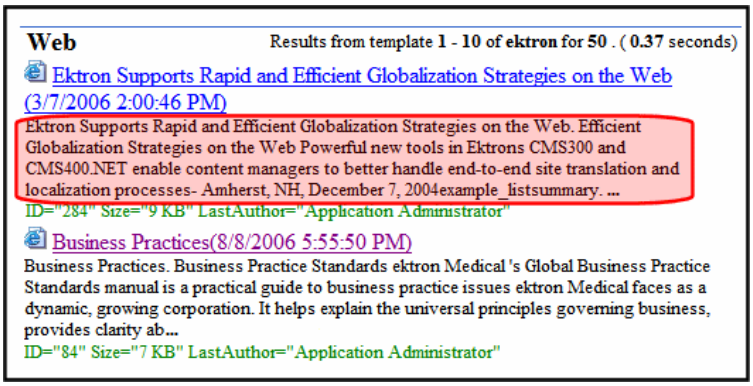
Results `[$PagingCurrentStartIndex]`.`[$PagingCurrentEndIndex]` of `[$ItemCount]` for item `[$SearchText]` (**`[$SearchDuration]`**).



[$SearchSummary]

Gets the first 300 characters from the body content and creates an abstract.

For content items that do not get indexed, such as images, the SearchSummary uses the image's title, summary, and metadata information.

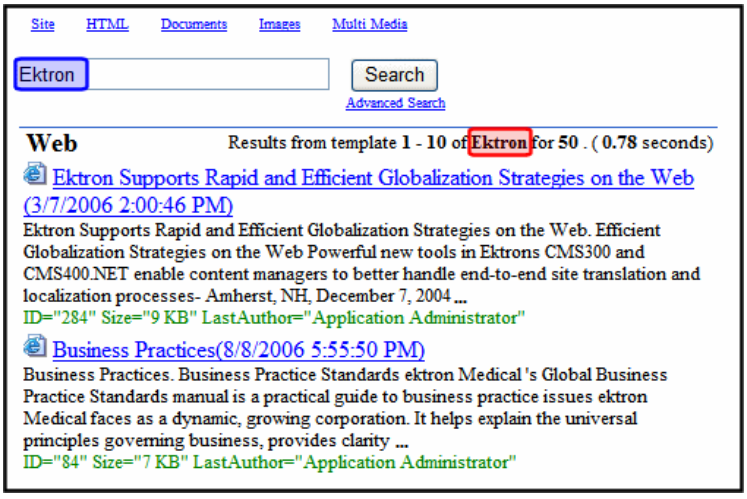


[$SearchText]

Displays the text for which a user is searching. This information is same as what a user entered in the search text box.

This variable is typically used in the following context:

Results `[$PagingCurrentStartIndex]`.`[$PagingCurrentEndIndex]` of `[$ItemCount]` for item **`[$SearchText]`** `([$SearchDuration])`.

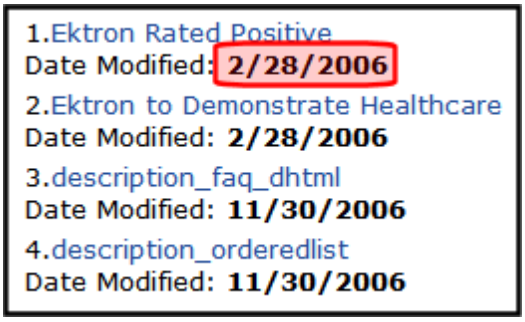


[$SERVER_NAME]

Displays the server name for the page on which this variable appears. For example, If this variable is applied to `http://www.example.com/demo.aspx`, the return is `www.example.com`.

[$ShortDateModified]

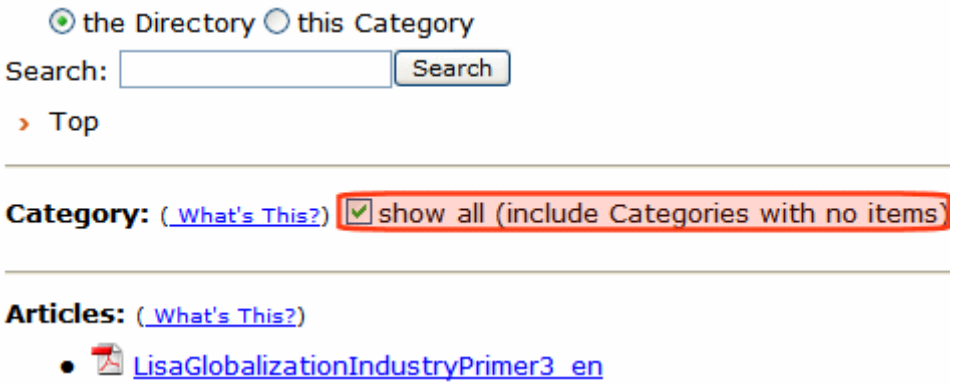
Displays the date the content was last modified. The variable show the date only. To show the Date and Time, use the *[$DateModified]* on page 2660 variable.




```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
        [$Index]. [$HyperLink] <br/>Date Modified: <b>[$ShortDateModified]</b>
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$ShowAllcategory]

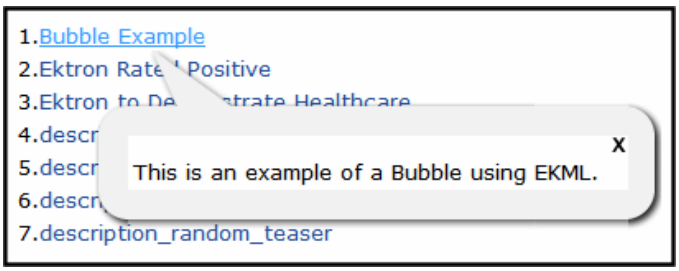
If a Directory server control's `EnableSearch` property is set to `true`, this variable adds the checkbox circled below to the right of the search field.



By default, the Directory server control only shows categories to which at least one content block is assigned. If you check this box, all categories appear, even those with no content assigned.

[$ShowBubble]

Calls the `<ekbubbleinfo>` tags and places the information contained within those tags in a pop-up bubble. This bubble is typically assigned to the `onclick` or `onmouseover` attribute in an `<a href="">` tag. See the following example EkML code.




```
<ekmarkup>
  <ekbubbleinfo>
    <table border="0">
      <tr>
        <td>[$Html]</td>
      </tr>
    </table>
  </ekbubbleinfo>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
        <tr>
          <td>
            [$Index].<a href="#" onmouseover="[$ShowBubble]">[$Title]</a>
          </td>
        </tr>
      </ekrepeat>
    </table>
```

```
  </ekoutput>
</ekmarkup>
```

[$ShowBubble(width,height)]

This is similar to `[$ShowBubble]`. It calls the `<ekbubbleinfo>` tags and places the information contained within those tags in a pop-up bubble. This variable lets you set the width and height of the bubble. For example, `[$ShowBubble(300,400)`. In this example, the first number represents the width. The second number represents the height.

If you enter a single number, it sets the width. The height of the bubble is then limited to the length of the content. For example, if you had a video that was formatted at 200 pixels wide and you wanted to launch it in a bubble, you would add the function as `[$ShowBubble(200)]`.

The minimum width for a bubble is 287. The minimum height is 101.

```
<ekmarkup>
  <ekbubbleinfo>
    <table border="0">
      <tr>
        <td>[$Html]</td>
      </tr>
    </table>
  </ekbubbleinfo>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
        <tr>
          <td>
           [$Index].<a href="#" onmouseover="[$ShowBubble(200)]">[$Title]</a>
          </td>
        </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$ShowContent('htmltagid')]

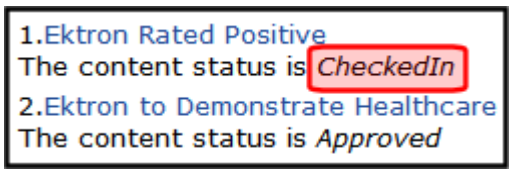
Calls the `<ekcontentinfo>` tags and places the information contained within those tags in the specified HTML tag ID. Replace the htmltagid with the ID of the tag. This tag is typically assigned to the `onclick` or `onmouseover` attribute in an `<a href="">` tag. See the example EkML code below.

```
<ekmarkup>
  <ekcontentinfo>
    <table border="0">
      <tr>
        <td>[$Html]</td>
      </tr>
    </table>
  </ekcontentinfo>
  <ekoutput>
```

```
    <table width="100%"  border="0">
      <ekrepeat>
        <tr>
          <td>
   [$Index].<a href="#" onmouseover="[$ShowContent('contarea')]">[$Title]</a>
          </td>
        </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$Status]

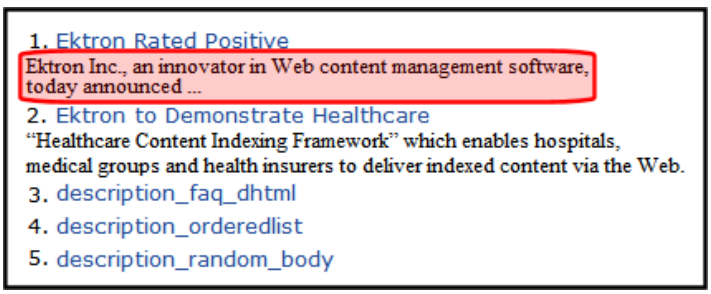
Displays the status of a content item. For example, Approved, Checked in, or Submitted for Approval. See also: Content Statuses.




```
<ekmarkup>
  <ekoutput>
   <table width="100%"  border="0">
    <ekrepeat>
      <tr>
       <td>
         [$Index]. [$Hyperlink]
          The content status is <i>[$Status]</i>
       </td>
      </tr>
    </ekrepeat>
   </table>
  </ekoutput>
</ekmarkup>
```

[$Teaser]

Displays the content item's summary information. A code example appears below.




```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
```

```
      <tr>
        <td>
          [$Index]. [$HyperLink] <br/> [$Teaser]
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$TemplateQuickLink]

Displays the Template Quicklink information assigned to the taxonomy item in the Workarea. When wrapped in an `<a href="">` tag, you can create a Hyperlink.

```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
         <a href="[$TemplateQuickLink]">[Title]</a><br/>[$Teaser]
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$Title]

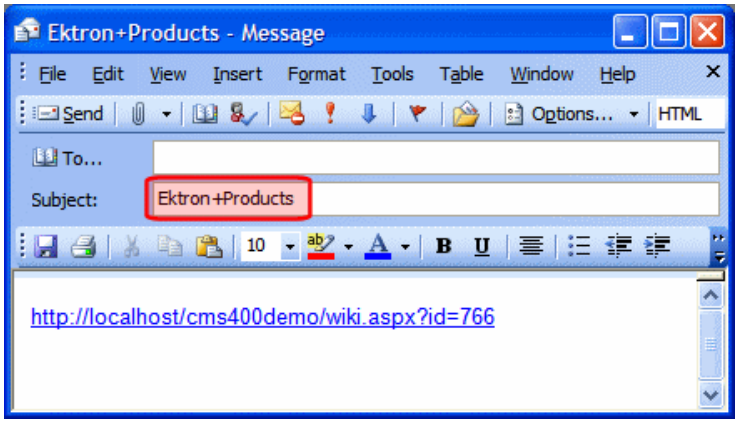
Displays the content item's title. Use this variable if you want to display the title as normal text. If you want to display the title as a hyperlink, use `[$HyperLink]`. See also: *[$HyperLink]* on page 2666. If you want to create a custom hyperlink with the title as the text of the hyperlink, use this property in conjunction with the `<a href="">` tag and the `[$QuickLink]` variable. See also: *[$QuickLink]* on page 2672.

```
<ekmarkup>
  <ekoutput>
    <table width="100%"  border="0">
      <ekrepeat>
      <tr>
        <td>
          [$Index]. <a href="[$QuickLink]">[$Title]</a>
        </td>
      </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$UrlEncode('str')]

Encodes the string information in the variable. This variable can be used to encode another EkML variable and place it in an email. For example, You want the title of the

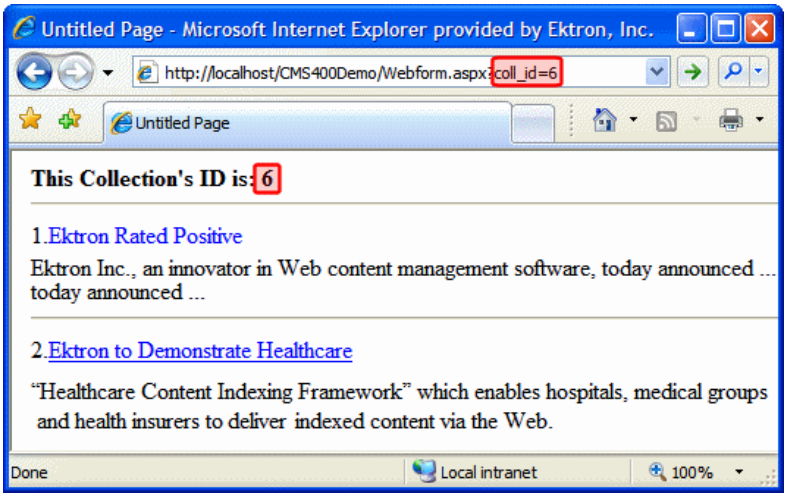
content block to appear as the subject of an email. A code example appears below.




```
<ekcontentinfo>
    <table border="0">
      <tr>
        <td>
          <strong>Title: </strong>
          <span style="color:red;">[$Title]</span>
          <a href="mailto:?subject=[$UrlEncode('[$Title]')]&body=
            http://[$SERVER_NAME]/CMS400Developer/wiki.aspx?id=
            [$ContentId]"> 
            <img src="WorkArea/images/application/icon_email.png"
              align="absmiddle" alt="[$Title]" border="0"/></a>
          <hr />
        </td>
      </tr>
      <tr>
        <td>[$Html]</td>
      </tr>
    </table>
  </ekcontentinfo>
```

[$UrlParam('paramname')]

Displays the value of a query string's parameter. For example, if you have a collection and want to display its ID, add the `[$UrlParam('coll_id')]` where you want the collection's ID to appear.




```
<ekmarkup>
  <ekbubbleinfo>
    <table border="0">
      <tr>
        <td>[$Html]</td>
      </tr>
    </table>
  </ekbubbleinfo>
  <ekoutput>
    <table width="100%"  border="0">
    <tr>
      <td>
        <b>This Collection's ID is: [$UrlParam('coll_id')]</b>
        <br/><hr/>
      </td>
    </tr>
      <ekrepeat>
        <tr>
          <td>
      [$Index].<a href="#" onclick="[$ShowBubble]">[$Title]</a>[$Teaser]<hr/>
          </td>
        </tr>
      </ekrepeat>
    </table>
  </ekoutput>
</ekmarkup>
```

[$UserName]

Display the Username of a user who left a comment on a message board.

```
<ekoutput>
    <div class="ContributionForm">
      <h4>[$NumberComments] Comments</h4>
      <ul>
        <ekrepeat>
          <li class="ekMessagePost">
            <div class="avatar">[$Avatar]</div>
            <div class="message">
              <div class="metaData">
                <span class="username">[$UserName]</span>
                <span class="time">[$DateCreated]</span>
              </div>
              <p>[$MessageText]</p>
              <ul class="commands">
                <li class="ekDeleteMessage">[$DeleteMessageLink]</li>
                <li class="ekApproveMessage">[$ApproveMessageLink]</li>
              </ul>
            </div>
          </li>
        </ekrepeat>
      </ul>
      [$AddCommentBox]
    </div>
  </ekoutput>
```

Index

E

F

G

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z