# Office AMS: Fields and Content Types

|  |  |
| --- | --- |
| Summary: | Applies to: |
| Provides scenarios for adding content types and fields to the host web site | * Office 365 Multi-Tenant (MT) * Office 365 Dedicated (D) * SharePoint 2013 on-premises |
| Solution: | Core.ContentTypesAndFields, version 1.0 |
| Author(S): | Vesa Juvonen, Bert Jansen (Microsoft) |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // THIS CODE IS PROVIDED \*AS IS\* WITHOUT WARRANTY OF  // ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING ANY  // IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR  // PURPOSE, MERCHANTABILITY, OR NON-INFRINGEMENT.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |

# General comments

This sample focuses on the following scenarios:

* **Scenario 1:** Create a new content type in the host web
* **Scenario 2:** Create a taxonomy field in the host web and wire it up to the taxonomy
* **Scenario 3:** Create a list and associate it with a content type
* **Scenario 4:** Create content types and fields in particular languages

# Scenario 1: Create New Content Type

This scenario demonstrates how we can create content types from the app web within the host web. This example uses the FieldAndContentTypeExtensions class in the OfficeAMS.Core.

When creating the content type, there is a check to determine if the content type already exists:

ctx.Web.ContentTypeExistsByName(txtContentTypeName.Text)

This will create the new content type

ctx.Web.CreateContentType(txtContentTypeName.Text, ctId, "Contoso Content Types");

It will then use the CreateField method to add a field to the content type which uses the FieldCollection.AddFieldAsXml method to create a new field.

public static Field CreateField(

this Web web,

Guid id,

string internalName,

FieldType fieldType,

string displayName,

string group,

string additionalXmlAttributes = "",

bool executeQuery = true)

Note that the **CreateField** method has the parameter *additionalXmlAttributes.* This is used to add other properties to the field XML. An example is in the **CreateTaxonomyField** method where ShowField=”1033”.

The *executeQuery* property enables one to bundle many fields and call ExecuteQuery at the end. It calls the web.Update() after each field, so if it fails, some of the fields will be created before completion.

It then adds the field to the content type.

ctx.Web.AddFieldToContentTypeByName(txtContentTypeName.Text, fieldId);

This is a process of ensuring that the field exists in the web, then creating a FieldLink.

FieldLinkCreationInformation fldInfo = new FieldLinkCreationInformation();

fldInfo.Field = field;

contentType.FieldLinks.Add(fldInfo);

contentType.Update(true);

web.Context.ExecuteQuery();

# Scenario 2: Create Taxonomy Field

This scenario creates a new taxonomy field and wires up the taxonomy field to the selected term group and term set.

It first checks to ensure that the field does not exist already

ctx.Web.FieldExistsByName(taxFieldName)

If it does exist, the field can be wired up to the term set via the **WireUpTaxonomyField** method. Which operates by getting the term group and term set and applying their IDs to the taxonomy field’s SSP ID and TermSetID properties.

// get the term group and term set

TermGroup termGroup = termStore.Groups.GetByName(mmsGroupName);

TermSet termSet = termGroup.TermSets.GetByName(mmsTermSetName);

web.Context.Load(termStore);

web.Context.Load(termSet);

web.Context.ExecuteQuery();

// set the SSP ID and Term Set ID on the taxonomy field

var taxField = web.Context.CastTo<TaxonomyField>(field);

taxField.SspId = termStore.Id;

taxField.TermSetId = termSet.Id;

taxField.Update();

web.Context.ExecuteQuery();

# Scenario 3: Create List and Apply Content Type

Now, it’s time to put it all together. The content type and fields can be added via the methods listed above and then the content type is applied to a new list.

ctx.Web.AddList(ListTemplateType.DocumentLibrary, txtListName.Text, false);

// Enable content types in list

List list = ctx.Web.GetListByTitle(txtListName.Text);

list.ContentTypesEnabled = true;

list.Update();

ctx.Web.Context.ExecuteQuery();

And after it applies the content type to the list, it sets the default content type on the list.

ctx.Web.SetDefaultContentTypeToList(txtListName.Text, contentTypeId);

This requires setting the UniqueContentTypeOrder on the list by getting all the content types and setting the sorted list on the root folder.

list.RootFolder.UniqueContentTypeOrder = newOrder;

# Scenario 4: Localizing Content Types and Fields

Localization of content types and fields requires that alternate languages be enabled on the site. To enable this, please view the blog entry on enabling localizations.

[Office365 – Multilingual content types, site columns and other site elements](http://blogs.msdn.com/b/vesku/archive/2014/03/20/office365-multilingual-content-types-site-columns-and-site-other-elements.aspx)

Localization of content types can be done via the **SetLocalizationForContentType** method.

ctx.Web.SetLocalizationForContentType("LitwareDoc2", "es-es", "Litware documento", "Litware documento");

Localization of fields can be done via the **SetLocalizationForField** method.

ctx.Web.SetLocalizationForField(fieldId, "es-es", "Field Name (es)", "Field Name (es)");

And setting the list name can be done via the **SetLocalizationLabelsForList** method.

ctx.Web.SetLocalizationLabelsForList(txtListName.Text, "es-es", "List name (es)", "List description (es)");