# Office AMS: Taxonomy Synchronization

|  |  |
| --- | --- |
| Summary: | Applies to: |
| This sample demonstrates how to Synchronize Terms across multiple term stores | * Office 365 Multi-Tenant (MT) * Office 365 Dedicated (D) * SharePoint 2013 on-premises |
| Solution: | Core.MMSSync, version 1.0 |
| Author: | Kimmo Forss, Frank Marasco (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

Have you ever wanted to move Taxonomy items from one term store to another? With on-premises implementations you can move your MMS database, but this takes some work. What about SharePoint Online? Obviously, we cannot move our database to Office 365. There is already an AMS (Core.MMS) sample that demonstrates how to perform basic operations with the term store. What if, you have use case where you only want to synchronize changes of a specific Term set? This sample takes the Core.MMS sample a step further, by using the ChangeInformation class in the Microsoft.SharePoint.Client.Taxonomy assembly.

http://blogs.msdn.com/b/frank\_marasco/archive/2014/06/29/synchronize-term-sets-with-the-term-store-csom.aspx

# SCENARIO 1

The first scenario, will take a Source Context and Term Group name and create a new Term Group in the target.

private void CreateTargetNewTermGroup(ClientContext sourceClientContext, ClientContext targetClientContext, TermGroup sourceTermGroup, TermStore targetTermStore)

{

try {

this.\_destinationTermGroup = targetTermStore.CreateGroup(sourceTermGroup.Name, sourceTermGroup.Id);

if(!string.IsNullOrEmpty(sourceTermGroup.Description))

{

this.\_destinationTermGroup.Description = sourceTermGroup.Description;

}

TermSetCollection \_sourceTermSetCollection = sourceTermGroup.TermSets;

if (\_sourceTermSetCollection.Count > 0)

{

foreach (TermSet \_sourceTermSet in \_sourceTermSetCollection)

{

sourceClientContext.Load(\_sourceTermSet,

set => set.Name,

set => set.Description,

set => set.Id,

set => set.Terms.Include(

term => term.Name,

term => term.Id),

term => term.Description,

term => term.Contact);

sourceClientContext.ExecuteQuery();

TermSet \_targetTermSet = \_destinationTermGroup.CreateTermSet(\_sourceTermSet.Name, \_sourceTermSet.Id, targetTermStore.DefaultLanguage);

if(!string.IsNullOrEmpty(\_sourceTermSet.Description))

{

\_targetTermSet.Description = \_sourceTermSet.Description;

}

foreach(Term \_sourceTerm in \_sourceTermSet.Terms)

{

Term \_targetTerm = \_targetTermSet.CreateTerm(\_sourceTerm.Name, targetTermStore.DefaultLanguage, \_sourceTerm.Id);

}

}

}

try

{

targetClientContext.ExecuteQuery();

targetTermStore.CommitAll();

}

catch

{

throw;

}

}

catch

{

throw;

}

}

# Scenario 2

This scenario will use the ChangeInformation class to process all changes in the source Term store to return all the changes that has occurred. First, create a TaxonomySession object.

DateTime \_startFrom = DateTime.Now.AddYears(-1);

Console.WriteLine("Opening the taxonomy session");

TaxonomySession \_sourceTaxonomySession = TaxonomySession.GetTaxonomySession(sourceClientContext);

TermStore sourceTermStore = \_sourceTaxonomySession.GetDefaultKeywordsTermStore();

sourceClientContext.Load(sourceTermStore);

sourceClientContext.ExecuteQuery();

Once you have created the TaxonomySession object we need to get the changes, we get the changes by creating a new Instance of ChangeInformation and set the start date. In this case, I’m getting all the changes from 1 year ago. I’m going to call the term store GetChanges method, which will return all the changes .

Console.WriteLine("Reading the changes");

ChangeInformation \_ci = new ChangeInformation(sourceClientContext);

\_ci.StartTime = \_startFrom;

ChangedItemCollection \_cic = sourceTermStore.GetChanges(\_ci);

sourceClientContext.Load(\_cic);

sourceClientContext.ExecuteQuery();

Once we invoke the GetChanges member this will return a ChangeItemCollection that  can be used to enumerate all the changes that have occurred in term store like we do below and take action based on the type of change that has occurred.

foreach (ChangedItem \_changeItem in \_cic)

{ ///ENUMERATE YOU’RE CHANGES

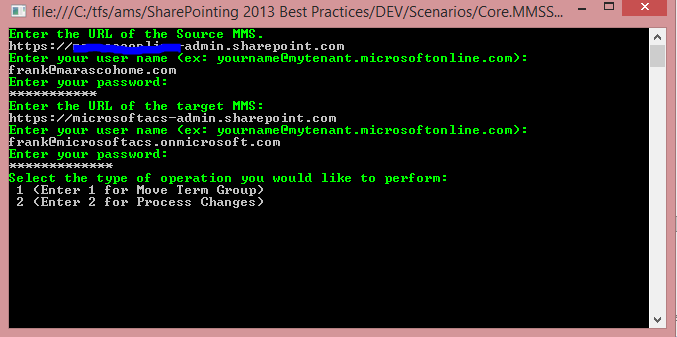
if (\_changeItem.ItemType == ChangedItemType.Group)

{

///PROCESS YOU’RE CHANGES

}  
}

# Running THE SAMPLE



Ensure, that the user has the appropriate permissions to the term store in both the source and target term stores, or you will get an exception.

# Dependencies

•Microsoft.SharePoint.Client.dll

•Microsoft.SharePoint.Client.Runtime.dll

•Microsoft.SharePoint.Client.Taxonomy.dll