Build a simple catalog management application for e-commerce

Maobing Jin (maobing@us.ibm.com)
Lead Architect, IBM MDM Industry Solutions and SaaS
IBM

Aditya Joag (aajoag@us.ibm.com)
Advisory Software Engineer, IBM MDM Industry Solutions
and SaaS
IBM

Rohit Gargate (rgargate@us.ibm.com)
Software Engineer, IBM MDM Industry Solutions and SaaS
IBM

Mohana Kera (mohakera@in.ibm.com)
Lead Developer, IBM MDM Industry Solutions and SaaS
IBM India

25 November 2014

In this tutorial, we will demonstrate a catalog management system for e-commerce solutions through an application on top of InfoSphere® MDM Collaborative Edition and WebSphere® Commerce Enterprise Edition. The application will provide a simple solution for managing catalog data through a web-based UI. It will model the catalog data for e-commerce products in MDM, provide collaborative environment for authoring catalog entries including products, SKUs, and bundles and kits, which can be organized and filtered in hierarchical catalog categories. This tutorial will guide readers through the process of developing their own solutions with the new features released in the Advanced Catalog Management Asset in InfoSphere MDM.

As MDM industry solution developers, we have been trying to solve the following problem: how to build a shared catalog management system for e-commerce solutions quickly. Authoring the product data collaboratively and provisioning the data services to other components in the system, for example, would be two of the key features in a catalog management application. In this tutorial, we will demonstrate an application implemented using Advanced Catalog Management (ACM) asset from IBM.

The asset is packaged in InfoSphere MDM 11.0.0 or later. It is developed on top of InfoSphere MDM Collaborative Edition (MDM) and WebSphere Commerce Enterprise Edition (WCS). Using
ACM, business users can author product data in MDM and persist it to WCS for servicing WCS applications, such as a storefront website.

**What you'll need to build a simple catalog management application**

You will need to install the Advanced Catalog Management (ACM) asset. ACM asset is packaged in the acm.zip file, available under the samples/acm folder of the MDM installation directory. Please see [ACM installation and configuration](#).

Before installing and configuring the ACM asset, you must install the following software:

1. InfoSphere MDM Collaborative Edition (MDM) — Creating a company for your catalog management solution, for example, naming the company as acm. Please see the details for [installing MDM](#).
2. WebSphere Commerce (WCS) — Install WebSphere Commerce with starter store archives. Please see the details of [WCS installation](#).
3. WebSphere Message Queue (MQ) — ACM uses Message Queue (MQ) for communication between MDM and WCS. A MQ installation needs to be available along with the WCS setup prior to ACM installation. Please see the details for [MQ installation and configuration](#).
4. File Transfer Protocol (FTP) software — ACM needs a FTP software for transferring files between MDM and WCS. A running installation of FTP software is required on the WCS setup prior to ACM installation. Please see the details for [FTP installation and configuration](#).

**Step 1. Create asset store and eSite store in WCS**

1. Open the WebSphere Commerce Administration Console and log in with your user and password — for example, wcsadmin and password.
2. On the next page, choose Site, click OK.
3. Open menu item Store Archives > Publish.
4. Choose Extended Sites from the View drop-down box.
5. Select store archive ExtendedSitesCatalogAssetStore-FEP.sar and click Next to specify the parameters:

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store directory</td>
<td>MyCatalogAssetStore</td>
</tr>
<tr>
<td>Store identifier</td>
<td>My Catalog Asset Store</td>
</tr>
<tr>
<td>Organization</td>
<td>Extended sites organization (root organization)</td>
</tr>
<tr>
<td>Inventory model</td>
<td>ATP</td>
</tr>
</tbody>
</table>

6. Click Next to go to the summary page, then click Finish to complete the publish.
For more information, please refer to the WCS detail instructions.

Now create an eSite Store with a sales catalog:

1. Follow the process for creating an asset store as above to publish a store archive `ExtendedSitesHub.sar`.
2. Open WebSphere Commerce Accelerator and log in with your user and password — for example, wcsadmin and password.
3. Select **Extended Sites Hub** and click **OK**.
4. On the pop-up window, open **Extended Sites > New Store**.
5. Specify the parameters:

   **Table 2. Parameter values for My eSite Store**

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store unique identifier</td>
<td>My eSite Store</td>
</tr>
<tr>
<td>Store display name</td>
<td>My eSite Store</td>
</tr>
<tr>
<td>Store description</td>
<td>eSite store for My Catalog Asset Store</td>
</tr>
<tr>
<td>Notification recipient email address</td>
<td><code>&lt;your email address&gt;</code></td>
</tr>
<tr>
<td>Default store currency</td>
<td>USD</td>
</tr>
<tr>
<td>Store organization</td>
<td>Extended sites hub organization</td>
</tr>
</tbody>
</table>

6. Go through the wizard to specify the parameters:

   **Table 3. Parameter values for eSite Store**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>storefrontassetstoretype</td>
<td><code>&lt;specify your storefront type&gt;</code></td>
</tr>
<tr>
<td>Catalog</td>
<td>My Catalog Asset Store</td>
</tr>
</tbody>
</table>

7. Click **Finish** to complete the process.
Finally, find the IDs for stores and the catalog that will be used in the next step:

2. Run the following SQL queries in the tool box to find IDs:

**Listing 1. Finding ID of My Catalog Asset Store**

```sql
select STOREENT_ID
from storeent
where IDENTIFIER='My Catalog Asset Store';
```

**Listing 2. Finding ID of Master Catalog in My Catalog Asset Store**

```sql
select CATALOG_ID
from storecat cat, storeent ent
where cat.STOREENT_ID=ent.STOREENT_ID and
  cat.MASTERCATALOG='1' and
  ent.IDENTIFIER='My Catalog Asset Store';
```

**Listing 3. Finding ID of My eSite Store**

```sql
select STOREENT_ID
from storeent
where IDENTIFIER='My eSite Store';
```

**Listing 4. Finding ID of Sales Catalog in My eSite Store**

```sql
select CATALOG_ID
from storecat cat, storeent ent
where cat.STOREENT_ID=ent.STOREENT_ID and
  cat.MASTERCATALOG='0' and
  ent.IDENTIFIER='My eSite Store';
```
Listing 5. Creating an Attribute Dictionary with ID for My Catalog Asset Store

```
insert into attrdict
(ATRDTICT_ID,STOREENT_ID,OPTCOUNTER) values (<unique number e.g. 10152>, <id of My Catalog Asset Store e.g. 10951>, 1);
```

Listing 6. Adding French language with ID -2 for My Catalog Asset Store

```
insert into storelang
(LANGUAGE_ID, STOREENT_ID) values (-2, <id of My Catalog Asset Store e.g. 10951>);
```

Step 2. Synchronize the configuration for catalogs and stores in MDM

You need to configure MDM with the stores and catalogs created on WCS.

ACM comes with node Master Catalog in Catalog Entry Categories hierarchy for the Master Catalog in WCS. You need to create a category node Catalog Entry Categories hierarchy for the sales catalog you created for the eSite store in WCS first:

1. On the left navigation pane, click the drop-down Please select a module to add. Select and add the hierarchy Catalog Entry Categories to the left navigation pane.
2. In the Catalog Entry Categories module, right-click on the hierarchy root node Catalog Entry Categories and add a Sales Catalog node.
3. On the right pane with single edit page, fill in the following information:

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>STRING</td>
<td>Sales catalog</td>
</tr>
<tr>
<td>Name</td>
<td>STRING</td>
<td>Sales catalog</td>
</tr>
<tr>
<td>MasterCatalog</td>
<td>FLAG</td>
<td>false</td>
</tr>
</tbody>
</table>

4. On single edit page, click Save to save the node data.
Now you need to provide IDs for each catalog in the lookup table Catalogs:

1. Open **Product Manager > Lookup Tables > Lookup Table Console**.
2. On the lookup table console, select **Catalogs** and click the button in the Action column to browse the table.
3. Select the rows for Master Catalog and Sales Catalog, and click **Edit Selected** to open bulk edit page.
4. On the Bulk Edit screen, fill the value in **Catalog Id** column for the selected entries.
5. Click **Save** to save the data.

**Figure 4. Confirming that catalog IDs in the lookup table are set up on your MDM system**

Create the node for My Catalog Asset Store in the stores hierarchy:

1. On the left navigation pane, click the drop-down **Please select a module to add**. Select and add the hierarchy **Stores** to the left navigation pane.
2. In the Stores module, right-click the hierarchy root node **Stores** and add a new **My Catalog Asset Store**.
3. On the right pane with single edit page, fill in the following information:

**Table 5. Attribute values for My Catalog Asset Store**

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
</table>

Figure 3. Confirming that sales catalog in the hierarchy is set up on your MDM system
4. On the single edit page, click **Save** to save the store data.

**Figure 5. Confirming that asset store in Stores hierarchy is set up on your MDM system**

Create a node for My eSite Store in stores hierarchy:

1. In the Stores module on the left navigation pane, right-click the **My Catalog Asset Store** node and add a **My eSite Store**.
2. On the right pane with single edit page, fill in the following information:

   **Table 6. Attribute values for My eSite Store**

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>STRING</td>
<td>My eSite Store</td>
</tr>
<tr>
<td>Store ID</td>
<td>STRING</td>
<td>&lt;the eSite store id retrieved from WCS in Step 1&gt;</td>
</tr>
<tr>
<td>Store type</td>
<td>STRING_ENUMERATION</td>
<td>eSite Store</td>
</tr>
<tr>
<td>Associated catalogs</td>
<td>LOOKUP_TABLE</td>
<td>Sales catalog</td>
</tr>
<tr>
<td>Attribute dictionary ID</td>
<td>LOOKUP_TABLE</td>
<td>&lt;NONE&gt;</td>
</tr>
</tbody>
</table>

3. On the single edit page, click **Save** to save the store data.
Step 3. Create attribute dictionary attributes

Attributes are modeled in an Attribute Dictionary hierarchy, which facilitates the use of selected attributes for different categories/groups of catalog entries.

There are two kinds of attributes, which must be created under the pre-defined defining attributes and descriptive attributes categories in the Attribute Dictionary hierarchy.

First, create a descriptive attribute:

1. On the left navigation pane, click drop-down Please select a module to add. Select and add the hierarchy Attribute Dictionary to the left navigation pane.
2. In the Attribute Dictionary module, expand the hierarchy root node Attribute Dictionary, then right-click on the Descriptive Attributes node and add a new Marketing Information node.
3. Create a new secondary spec Highlights Spec with one attribute Highlights:

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highlights</td>
<td>STRING</td>
<td>MAX_OCC:1</td>
</tr>
</tbody>
</table>

4. Click Manage Specs > Add Spec button to add spec Highlights Spec as Item Spec to category Marketing Information in Attribute Dictionary hierarchy.
5. Click Save to save the category node.
Create a defining attribute:

1. In the Attribute Dictionary module, right-click on the **Defining Attributes** node and add a new **Customer Information** node.
2. On the right pane, click **Manage Specs > Add Spec** to open a pop-up window for adding secondary spec.
3. On the pop-up window, under the **Create Simple Spec** section, enter **Customer types Spec** for spec name in the text box.
4. Click **Create** to create new secondary spec with one attribute **Customer types**

   **Table 8. Customer types spec**

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer types</td>
<td>STRING_ENUMERATION</td>
<td>Enumeration values:{Senior, Adult, Junior, Minor, Baby}</td>
</tr>
</tbody>
</table>

5. Click **Save** to save the category node.

Finally, you need to publish attributes in Attribute Dictionary to WCS:

1. Open **Custom Tools > ACM Attribute Dictionary Export** to gather all attributes and publish them to the corresponding Attribute Dictionary in the My Catalog Asset Store in WCS.
Publishing attributes from MDM to WCS is implemented using web services. You can read the code snippet below for more implementation details.

**Listing 7. ExportAttributeDictionary.java**

```java
void exportAttributeDictionary()
{
    List<IntegrationStatus> statusList = cusTool.exportAttributes();
    cusTool.exportAttributeGroupsAndRelationships();
}
```

**Step 4. Create a catalog group with a product and a SKU in Master Catalog**

Catalog groups are modeled in the Catalog Entry Categories hierarchy. As a category node in the hierarchy, you can create product or SKU under a catalog group.

First, create a catalog group:

1. In the **Catalog Entry Categories** module on the left navigation pane, expand the hierarchy root of the **Catalog Entry Categories**.
2. Right-click the **Master Catalog** node and select **Add Category** or **Add Group** (if multiple domain entity-enabled).
3. Create a new Cell Phones category by filling out the following attributes in the single edit pane:

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>STRING</td>
<td>Cell Phones</td>
</tr>
<tr>
<td>Name</td>
<td>STRING</td>
<td>Cell Phones</td>
</tr>
<tr>
<td>MasterCatalog</td>
<td>FLAG</td>
<td>false</td>
</tr>
<tr>
<td>Search engine optimization/URL keyword</td>
<td>STRING</td>
<td>cellphones</td>
</tr>
</tbody>
</table>

4. On the single edit page, click **Save** to save the new catalog group.
5. Click **Checkout** and select **Catalog Group Export Collaboration Area** to publish catalog group Cell Phones from MDM to WCS.

6. Add the module **Catalog Group Export Collaboration Area** to the left navigation pane and expand the collaboration area.

**Figure 11. Confirming that catalog group in the hierarchy is exported from MDM to WCS**

7. Click on **Cell phones** under **Review Completed Catalog Groups** to open the catalog group on the single edit page.

8. Click **Done** on the single edit page to complete the workflow.

Publishing catalog group from MDM to WCS is implemented using Web Services. The code below contains more implementation details.

**Listing 8. CatalogGroupExportWorkflowStepFunction.java**

```java
public void out(WorkflowStepFunctionArguments inArgs)
{
    try
    {
        IntegrationStatus status = null;
```
// data from workflow step
PIMCollection<CollaborationCategory> categoryset = inArgs.getCategories();
CollaborationArea collaborationArea = inArgs.getCollaborationStep().getCollaborationArea();
Workflow workflow = collaborationArea.getWorkflow();
CollaborationStep step = inArgs.getCollaborationStep();
String stepPath = step.getName();

// split categoryset into new and existing groups
List<MdmCategory> newMdmEntryList = new ArrayList<MdmCategory>();
List<MdmCategory> existingMdmEntryList = new ArrayList<MdmCategory>();
Context currentContext = PIMContextFactory.getCurrentContext();
boolean hasURLKeyword = false;
boolean hasSEOProperties = false;

int storeId = -1;
int catalogId = -1;
boolean isMasterCatalog = false;
int attributeDictionaryId = -1; //ACMConfiguration.DEFAULT_WCS_ATTRIBUTE_DICTIONARY_IDENTIFIER;

Hierarchy storeHierarchy =
currentContext.getHierarchyManager().getHierarchy(ACMConfiguration.HIERARCHY_CATALOG_ENTRY_STORE);
PIMCollection<Category> stores = storeHierarchy.getCategories();
LookupTable catalogsLkp =
currentContext.getLookupTableManager().getLookupTable(ACMConfiguration.MDM_CATALOGS_LOOKUP_TABLE_NAME);

for (Iterator<CollaborationCategory> iterator = categoryset.iterator(); iterator.hasNext(); )
{
  CollaborationCategory collabCat = iterator.next();
  String catalogGroupIdentifier = collabCat.getPrimaryKey();
  if (catalogGroupIdentifier == null)
  {
    continue;
  }

  //make sure WCS Catalogs are not being exported.
  int[] levels = collabCat.getLevels();
  if (levels[0] == 1)
  {
    throw new UnsupportedOperationException(resourceLocator.getResourceString
      ("acm.export.jobs.catalogCantBeExported", new Object[] {collabCat.getName()});
  }

  MdmCategory mdmCategory = new MdmCategory(ACMConfiguration.HIERARCHY_CATALOG_ENTRY_CATEGORIES,
catalogGroupIdentifier, currentContext);
  if (!mdmCategory.hasAttributeValue(ACMConfiguration.ATTRIBUTE_PATH_CATALOG_GROUP_WCSID))
  {
    newMdmEntryList.add(mdmCategory);
  }
  else
  {
    existingMdmEntryList.add(mdmCategory);
  }

  if (mdmCategory.isAttributePresent(ACMConfiguration.ATTRIBUTE_PATH_CATALOG_GROUP_SEO_URLKEYWORD))
  {
    hasURLKeyword = true;
  }

  if (mdmCategory.isAttributePresent(ACMConfiguration.ATTRIBUTE_PATH_CATALOG_GROUP_SEO_PAGETITLE) ||
      mdmCategory.isAttributePresent(ACMConfiguration.ATTRIBUTE_PATH_CATALOG_GROUP_SEO_IMAGE_ALTTEXT) ||
      mdmCategory.isAttributePresent(ACMConfiguration.ATTRIBUTE_PATH_CATALOG_GROUP_SEO_META_DESC))
  {
    hasSEOProperties = true;
  }

  //The category that is being exported may not be a 'catalog' category(might be a sub-category of a catalog).
  //Traverse up to the parent which is at level 1 to get the store association and other info ...
  CollaborationCategory catalogCat = getCatalogCat(collabCat);
//retrieve the storeId, catalogId and attribDictId from the store which is using this catalog group
//As of now, the following code needs to get run for each category. However, this can get removed
//once the attribute Associated Store under the catalog group is populated and can be read directly.
//Iterate thru' each store under Stores Hierarchy and find out which store has this catalog group as the
associated catalog
    String catalogCatName = catalogCat.getName();
    for (Category store : stores)
    {
        boolean storeMatchFound = false;
        List<? extends AttributeInstance> assocCatalogNameInsts =
            store.getAttributeInstance(ACMConfiguration.ATTRIBUTE_PATH_ASSOCIATED_CATALOGS).getChildren();
        if (assocCatalogNameInsts != null)
        {
            for (AttributeInstance assocCatalogNameInst : assocCatalogNameInsts)
            {
                Object assocCtgNameVal = assocCatalogNameInst.getValue();
                if (assocCtgNameVal == null) continue;  //this can happen if the associated catalog was deleted
                when association is present
                if (assocCtgNameVal.toString().equals(catalogCatName))
                {
                    storeMatchFound = true;
                    int tempStoreId =
                        Integer.parseInt(store.getAttributeValue(ACMConfiguration.ATTRIBUTE_PATH_STORE_STORE_ID).toString());
                    //only catalog groups with same storeId can be exported together.
                    if (storeId != -1 && storeId != tempStoreId) {
                        throw new UnsupportedOperationException(resourceLocator.getResourceString
                            ("acm.export.jobs.onlyOneStoreSupportedForCatalogGroup", new Object[] {""+storeId,
                            ""+tempStoreId");
                    }
                    storeId = tempStoreId;
                    //read the catalog id for the catalog group from Catalogs lookup. Throw exception if it wasn't
                    set.
                    LookupTableEntry assocCtgEntry = catalogsLkp.getLookupTableEntry(catalogCatName);
                    if (assocCtgEntry != null)
                    {
                        Object catalogIdVal =
                            assocCtgEntry.getAttributeValue(Catalogs.ATTRIBUTE_PATH_LOOKUPTABLE_CATALOGS_CATALOG_ID);
                        if (catalogIdVal == null) {
                            String errMsg = resourceLocator.getResourceString
                                ("acm.export.jobs.catalogIdNotConfiguredForCatalog", new Object[]
                    {ACMConfiguration.MDM_CATALOGS_LOOKUP_TABLE_NAME, catalogCatName});
                            logger.logInfo(errMsg);
                            throw new PIMInternalException(errMsg);
                        }
                        catalogId = Integer.parseInt(catalogIdVal.toString());
                    }
                    Object isMasterCatalogVal =
                        catalogCat.getAttributeValue(ACMConfiguration.ATTRIBUTE_PATH_CATALOG_ENTRY_CATEGORIES_MASTERCATALOG);
                    if (isMasterCatalogVal != null) {
                        isMasterCatalog = isMasterCatalogVal.toString().equals("true") ? true : false;
                    }
                    Object attrDictVal =
                        store.getAttributeValue(ACMConfiguration.ATTRIBUTE_PATH_STORE_ATTRIBUTE_DICTIONARY);
                    if (attrDictVal != null)
                    {
                        AttributeDictionaryRegistry attrDictReg = new AttributeDictionaryRegistry();
                        attributeDictionaryId = attrDictReg.getAttributeDictionaryId(attrDictVal.toString());
                    }
                    if (storeMatchFound) break;
                }
            }
        }
    }
    if (storeMatchFound) break;
if (storeId < 0) {
    String errMsg = resourceLocator.getResourceString("acm.export.jobs.invalidStoreIdFound");
    logger.logInfo(errMsg);
    throw new PIMInternalException(errMsg);
}

WCSBusinessContext wcsBusinessContext = new WCSBusinessContext(
    ACMConfiguration.WCS_CONTEXT_USERNAME,
    ACMConfiguration.WCS_CONTEXT_PASSWORD,
    storeId,
    catalogId,
    isMasterCatalog,
    attributeDictionaryId);

// create service instance
CatalogGroupWorkflowStepExport catGroupWflStepService = new CatalogGroupWorkflowStepExport(categoryset,
collaborationArea, workflow, step, stepPath);

// export new catalog groups
if (!newMdmEntryList.isEmpty()) {
    BusinessObjectTransaction catalogGroupCreateTransaction = new CatalogGroupExporter(
        new CatalogGroupTransformer(),
        new CatalogGroupCreateWSLoader(ACMConfiguration.WCS_WS_ENDPOINT_URL,
        wcsBusinessContext));
    status = catGroupWflStepService.execute(newMdmEntryList, catalogGroupCreateTransaction, new
    ACMSoapResponseXmlHandler());
    updateMdmCategoryAttributesOnExport(newMdmEntryList, status);
    logger.logInfo(status);

    if (hasSEOProperties) {
        updateCatalogGroupSEOProperties(newMdmEntryList, wcsBusinessContext, catGroupWflStepService);
    }

    if (hasURLKeyword) {
        updateCatalogGroupSEOURL(newMdmEntryList, wcsBusinessContext, catGroupWflStepService);
    }
} else if (!existingMdmEntryList.isEmpty()) {
    // export existing catalog groups
    BusinessObjectTransaction catalogGroupUpdateTransaction = new CatalogGroupExporter(
        new CatalogGroupTransformer(),
        new CatalogGroupChangeWSLoader(ACMConfiguration.WCS_WS_ENDPOINT_URL,
        wcsBusinessContext));
    status = catGroupWflStepService.execute(existingMdmEntryList, catalogGroupUpdateTransaction, new
    ACMSoapResponseXmlHandler());
    updateMdmCategoryAttributesOnExport(existingMdmEntryList, status);
    logger.logInfo(status);

    if (hasSEOProperties) {
        updateCatalogGroupSEOProperties(existingMdmEntryList, wcsBusinessContext, catGroupWflStepService);
    }

    if (hasURLKeyword) {
        updateCatalogGroupSEOURL(existingMdmEntryList, wcsBusinessContext, catGroupWflStepService);
    }
}
}

catch(Exception e) {
    logger.logError(e.getMessage());
    throw new PIMInternalException(e.getMessage());
Create a product and a SKU under the catalog group:

1. On the left navigation pane, select right-click the **Cell Phones** category and select **Add Item** or **Add Entry** (if multiple domain entity-enabled).
2. On the single edit page of the right pane, fill out the following attributes:

   **Table 10. Attribute values for Smartphone product**

<table>
<thead>
<tr>
<th>Tab</th>
<th>Attribute name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage entity</td>
<td>Code</td>
<td>STRING</td>
<td>Smartphone</td>
</tr>
<tr>
<td>Manage entity</td>
<td>Name</td>
<td>STRING</td>
<td>Smartphone</td>
</tr>
<tr>
<td>Categories</td>
<td>Category</td>
<td>MAPPING</td>
<td>Catalog Entry Types Product (selected from Category Mapping Editor)</td>
</tr>
<tr>
<td>Categories</td>
<td>Category</td>
<td>MAPPING</td>
<td>Stores/My Catalog Asset Store (selected from Category Mapping Editor)</td>
</tr>
</tbody>
</table>

3. Click **Save** to save the product. See the following code for implementation details.

   **Listing 9. CatalogEntryRepositoryPostSaveFunction.java**

   ```java
   package com.ibm.mdm.acm.extensionpoints;
   
   import com.ibm.mdm.acm.model.CatalogEntryRepository;
   import com.ibm.mdm.integration.utils.ResourceLocator;
   import com.ibm.mdm.acm.model.CatalogEntryRepository;
   import com.ibm.pim.catalog.item.Item;
   
   import com.ibm.pim.collaboration.CollaborationItem;
   import com.ibm.pim.context.PIMContextFactory;
   import com.ibm.pim.extensionpoints.PostCategorySaveFunctionArguments;
   import com.ibm.pim.extensionpoints.PostCollaborationCategorySaveFunctionArguments;
   import com.ibm.pim.extensionpoints.PostCollaborationItemSaveFunctionArguments;
   import com.ibm.pim.extensionpoints.PostItemSaveFunctionArguments;
   import com.ibm.pim.extensionpoints.PostCategorySaveFunctionArguments;
   import com.ibm.pim.extensionpoints.PostItemSaveFunctionArguments;
   import com.ibm.pim.utils.Logger;
   
   public class CatalogEntryRepositoryPostSaveFunction implements 
   com.ibm.pim.extensionpoints.PostSaveFunction 
   {
   private static Logger logger = 
   PIMContextFactory.getCurrentContext().getLogger(CatalogEntryRepositoryPostSaveFunction.class.getName());
   
   private CatalogEntryRepository catalogEntryRepository = new CatalogEntryRepository();

   @Override
   public void postsave(PostItemSaveFunctionArguments inArgs)
   {
   Item item = inArgs.getItem();
   catalogEntryRepository.validate(item);
   catalogEntryRepository.updateXSLMapping(item);
   catalogEntryRepository.manageSequenceGroupsForCatalogEntry(item);
   }

   @Override
   public void postsave(PostCategorySaveFunctionArguments inArgs)
   {
   }
ResourceLocator resourceLocator = ResourceLocator.getSystemResourceLocator("com.ibm.mdm.acm.nls.resources.Resources");
String msg = resourceLocator.getResourceString("acm.extensionpoints.itemExtensionPointError");
logger.logInfo(msg);
throw new UnsupportedOperationException(msg);
}

@Override
public void postsave(PostCollaborationItemSaveFunctionArguments inArgs)
{
    CollaborationItem collabItem = inArgs.getCollaborationItem();
catalogEntryRepository.validate(collabItem);
catalogEntryRepository.updateXSLMapping(collabItem);
}

@Override
public void postsave(PostCollaborationCategorySaveFunctionArguments inArgs)
{
    ResourceLocator resourceLocator = ResourceLocator.getSystemResourceLocator("com.ibm.mdm.acm.nls.resources.Resources");
    String msg = resourceLocator.getResourceString("acm.extensionpoints.itemExtensionPointError");
    logger.logInfo(msg);
    throw new UnsupportedOperationException(msg);
}

4. On the single edit page, click **Checkout** and select **Generate SKU** to create a new SKU for the product Smartphone. See the following code for implementation details.

**Listing 10. GenSKU.script**

```javascript
var sErrMsg = null;
catchError(sErrMsg) {
    log("Entered OUT function of create SKU");
    var sCtgSourceName = colArea.getColAreaSrcContainer().getCtgName();
    var scrSKU = getScriptByPath("/scripts/triggers/ACM.SKU");
    var createSKU = scrSKU.getFunctionByName("createSKU");

    forEachEntrySetElement(entrySet, entry) {
        var sKey = entry.getPrimaryKey();
        log("Create SKU for: " + checkString(sKey, ""));
        createSKU.invoke(sKey, sCtgSourceName);
        entry.setExitValue("SUCCESS");
    }

    log("Finished OUT function of create SKU");
}
if (sErrMsg != null) {
    log("Error: " + sErrMsg);
}
```

5. Refresh the **Catalog Entry Repository** module on the left navigation pane until you find the generated SKU.
Finally, you can publish catalog entries to WCS:

1. Open **Product Manager > Selections > New Static Selection** to create a Cellphones selection.

2. Open **Custom Tools > ACM Catalog Entry Selective Export**.

3. On the **ACM Selective Export** page, select the **Cellphones** row, then click **Export** to bring up the export wizard.

4. Select **My Catalog Asset Store** from the store drop-down, then click **Export** on the wizard.
Figure 14. Confirming that the exporting product and SKU in the repository is kicked off in your MDM system

Listing 11. AbstractBatchLoader.java

```java
public void load(List<BusinessObject> businessObjectList) throws BusinessObjectLoaderException {
    logger.logDebug(resourceLocator.getResourceString("integration.common.loaders.loadStarted"));
    try {
        // compose soap message
        String dataFilePath = collectBusinessDataInFile(businessObjectList);
        logger.logDebug(resourceLocator.getResourceString("integration.common.loaders.loadDataFile", new Object[] {dataFilePath}));
        // send data and notify
        loadData(dataFilePath);
        logger.logDebug(resourceLocator.getResourceString("integration.common.loaders.loadSubmitted"));
    } catch (Exception e) {
        logger.logInfo(resourceLocator.getResourceString("integration.common.loaders.loadException", new Object[] {e.getMessage()}));
        throw new BusinessObjectLoaderException(BusinessObject.convertToMdmObjectList(businessObjectList), e);
    }
}
```

To verify the result of publishing catalog entries:

1. Open IBM Management Center for WebSphere Commerce.
2. Open Management Center Tools > Catalogs to add a Catalogs tab to the page.
3. On the Catalogs tab, click the drop-down box on the right-upper corner and select My Catalog Asset Store to open the store on the page.
4. On the left navigation pane, click the drop-down and select Master Catalog.
5. On the left navigation pane, expand My Catalog Asset Store and browse to the Cell Phones catalog group. The Smartphone product will be listed under Cell Phones and its SKU will be listed in the right pane.
Step 5. Create sales categories with selected product and SKU

My eSite Store was created for a seller to select the subset of the catalog that will be presented and adjusts the descriptions as necessary.

To construct a sales catalog in My eSite Store, create sales categories under the Sales Catalog node in Catalog Entry Categories hierarchy:

1. On the Catalog Entry Categories module of the left navigation pane, expand the hierarchy and browse to the node Sales Catalog.
2. Right-click on node Sales Catalog and select Add Category or Add Catalog Group (if multiple domain entity-enabled) to add a new sales category Cell phones for sale, with the following values:

<table>
<thead>
<tr>
<th>Tab</th>
<th>Attribute name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage category</td>
<td>Code</td>
<td>STRING</td>
<td>Cell phones for sale</td>
</tr>
<tr>
<td>Manage category</td>
<td>Name</td>
<td>STRING</td>
<td>Cell phones for sale</td>
</tr>
<tr>
<td>Manage category</td>
<td>Short description</td>
<td>STRING</td>
<td>for summer sales</td>
</tr>
<tr>
<td>Manage category</td>
<td>Long description</td>
<td>RICH TEXT</td>
<td>for summer sales</td>
</tr>
<tr>
<td>Manage category</td>
<td>MasterCatalog</td>
<td>FLAG</td>
<td>false</td>
</tr>
<tr>
<td>Search engine optimization</td>
<td>URL keyword</td>
<td>STRING</td>
<td>cellphonesforsale</td>
</tr>
</tbody>
</table>

3. Click Save to save the Sales category.
Figure 16. Confirming that the Sales category in the repository is created in your MDM system

4. Follow the same process used in Step 5 to publish the catalog group Cell phones for sale to WCS.

Select product and SKU for sales catalog:

1. On the Catalog Entry Repository module of the left navigation pane, expand the Catalog Entry Categories hierarchy to the Cell Phones category under Master Catalog.
2. Open product Smartphone under single-edit page on the right pane and go to the Categories tab to map the product to the additional categories below:

<table>
<thead>
<tr>
<th>Tab</th>
<th>Attribute name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categories</td>
<td>Category</td>
<td>MAPPING</td>
<td>Catalog entry categories/Cell phones for sale (selected from Category Mapping Editor)</td>
</tr>
<tr>
<td>Categories</td>
<td>Category</td>
<td>MAPPING</td>
<td>Stores/My eSite Store (selected from Category Mapping Editor)</td>
</tr>
</tbody>
</table>

3. Click Save to save the changes.

Figure 17. Confirming product and SKU in the repository are mapped to Sales Catalog in your MDM system

4. Follow the same steps above to map the SKU Smartphone-000 to the two additional categories as shown in the table above.

In My eSite Store, you can override the description of a SKU through the DescriptionOverride support:
1. On the Catalog Entry Repository module of the left navigation pane, expand the **Catalog Entry Categories** hierarchy and browse to the category **Cell phones for sale** under the parent category **Sales Catalog**.

2. Open SKU **Smartphone-000** under the category **Cell phones for sale** to the single-edit page on the right pane.

3. On the single edit page, go to the tab **Description Override** and provide the following attribute values.

   **Table 13. Additional DescriptionOverride attribute values for Smartphone-000 SKU**

<table>
<thead>
<tr>
<th>Tab</th>
<th>Attribute name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description Override</td>
<td>Name</td>
<td>STRING</td>
<td>Smartphone with special offer</td>
</tr>
<tr>
<td>Description Override</td>
<td>Short description</td>
<td>STRING</td>
<td>Smartphone with the HD and camera.</td>
</tr>
<tr>
<td>Description Override</td>
<td>Long description</td>
<td>RICH TEXT</td>
<td>Smartphone with the 1080p high-resolution and 12MP camera.</td>
</tr>
</tbody>
</table>

4. Click **Save** to save the changes.

**Figure 18. Confirming that product and SKU in the repository are mapped to Sales Catalog in your MDM system**

Finally, you can publish the SKU with the overridden descriptions in Sales catalog to WCS:

1. Open menu **Product Manager > Selections > New Static Selection** to create a new selection **Cellphones for sale**.
Figure 19. Confirming that selection is created in your MDM system

2. Open menu **Custom Tools > ACM Catalog Entry Selective Export**.
3. On the ACM Selective Export page, select **Cellphones for sale** and click **Export** to call the export wizard.
4. Select **My eSite Store** from the Store drop-down box and click **Export** on the wizard.

Figure 20. Confirming that exporting product and SKU in the repository is kicked off in your MDM system

See the code below for these conditions invoked through a JSP:

**Listing 12. ExportCatalogEntryDescriptionOverride.java**

```java
if( ((String)store.getAttributeValue(STORE_TYPE_PATH)).equalsIgnoreCase(ESITE_STORE) ) {
    DescriptionOverrideDataLoadJob doDataLoad = new DescriptionOverrideDataLoadJob(selection, store_id, storeName);
    doDataLoad.run();
} else {
    DataLoadJob dataLoad = new DataLoadJob(selection, store_id, storeName);
    dataLoad.run();
}
```

You can confirm the result by following the same process listed in Step 5.
Step 6. Enable multiple languages

Catalog attributes for description of catalog entries and groups plus the user-defined attributes in Attribute Dictionary are locale-sensitive. You can enable those attributes for the selected languages from the following list.

Table 14. Supported languages

<table>
<thead>
<tr>
<th>Language name</th>
<th>Language ID</th>
<th>Language code</th>
</tr>
</thead>
<tbody>
<tr>
<td>English (US)</td>
<td>-1</td>
<td>en_US</td>
</tr>
<tr>
<td>French</td>
<td>-2</td>
<td>fr_FR</td>
</tr>
<tr>
<td>German</td>
<td>-3</td>
<td>de_DE</td>
</tr>
<tr>
<td>Italian</td>
<td>-4</td>
<td>it_IT</td>
</tr>
<tr>
<td>Spanish</td>
<td>-5</td>
<td>es_ES</td>
</tr>
<tr>
<td>Brazilian Portuguese</td>
<td>-6</td>
<td>pt_BR</td>
</tr>
<tr>
<td>Simplified Chinese</td>
<td>-7</td>
<td>zh_CN</td>
</tr>
<tr>
<td>Traditional Chinese</td>
<td>-8</td>
<td>zh_TW</td>
</tr>
<tr>
<td>Korean</td>
<td>-9</td>
<td>ko_KR</td>
</tr>
<tr>
<td>Japanese</td>
<td>-10</td>
<td>ja_JP</td>
</tr>
<tr>
<td>Russian</td>
<td>-20</td>
<td>ru_RU</td>
</tr>
<tr>
<td>Polish</td>
<td>-22</td>
<td>pl_PL</td>
</tr>
</tbody>
</table>

For example, to enable French for the description attributes of product Smartphone:

1. Open menu **Data Model Manager > Specs/Mappings > Spec Console**.
2. Click **Edit** on the Catalog Entry Spec.
3. On the Details for Catalog Entry Spec page, select **French** from **Available Locales** and add it to **Selected Locales**.
4. Browse to attribute **Catalog Entry Spec/General information/Keyword** and click to open it.
5. On the Details for Keyword page, check the **Localized** box.

6. Click **Save** to save the changes so far on the spec.

   **Figure 22. Confirming that spec for Catalog Entry repository is localized in your MDM system**

![Localization image](image)

7. Open **Data Model Manager > Attribute Collections > Attribute Collection Console**.

8. Open the **Manage Entry** attribute collection.

9. On the Attribute Collection page, enter **Catalog Entry Spec** in the **Spec Name** field and **Keyword** in the **Attribute Path** field, then click the search arrow button.

10. Select the found attribute **Keyword** and click **Selected** to add this attribute to the collection.

   **Figure 23. Confirming that spec for Catalog Entry repository is localized in your MDM system**

![Attribute Collection image](image)

With the attribute enabled for French language support, you provide the locale specific value for the attribute.

1. From the left navigation pane, open the product Smartphone in the single-edit page on the right pane.

2. Enter **mot-clé** in the **Keyword\fr_FR** field.

3. Click **Save** to save the changes.
Figure 24. Confirming that localized values for Catalog Entry are saved in your MDM system

4. Follow the same process in Step 5 to publish the catalog entries to WCS again by using the Cellphones selection on the custom ACM Catalog Entry Selective Export tool page.

Conclusion

In this tutorial, we used the ACM asset from InfoSphere MDM to build a smarter commerce solution on top of two IBM products: InfoSphere MDM and WebSphere Commerce. We recommend customers to leverage the asset and customize it for your own business needs.

For more information about using the ACM asset, you can refer to the official product documentation.
Resources

Learn

- Learn more about WebSphere Commerce.
- Learn more about Big data and analytics.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
- Visit the developerWorks Information Management zone to find more resources for DB2 developers and administrators.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Follow developerWorks on Twitter.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.

Get products and technologies

- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, or use a product in a cloud environment.

Discuss

- Get involved in the developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.
About the authors

Maobing Jin

Maobing Jin is a senior software architect in the InfoSphere MDM division at IBM. He has been working on product information management product development since 2004, designed and implemented the product information search engine with business object query language and programming interfaces; enhanced the workflow engine with improved transaction management; as well as other features, such as MDM web services and single sign-on. Recently, Jin has been leading the development on MDM industry solution development and SaaS offering, delivered product catalog management and financial services solutions, plus a robust synchronization framework for MDM product domain.

Aditya Joag

Aditya Joag is working as an advisory software engineer in Software Design and Development of the InfoSphere MDM Collaboration Server. He has in-depth experience working with databases, cloud computing, and artificial intelligence.

Rohit Gargate

Rohit Gargate is working as a software developer on the InfoSphere Master Data Management team at IBM. Currently, he is working on building solutions for MDM Collaborative Edition using Java, JSP/Dojo, and XML.

Mohana Kera

Mohana Kera has more than 15 years of in-depth experience in architecture, design, development and implementation of software solutions. He has been working at IBM for more than 7 years, primarily on InfoSphere MDM Collaboration Server. He has in-depth experience with all aspects of the product and was the development lead for Java API interfaces of the system. He has published several technotes, provided customer training on Java APIs, and has actively worked in developing the content for IBM Information Center around MDM Collaboration Server Java APIs.