

IBM WebSphere Commerce Feature Pack 3.01 Lab exercise

## Catalog Web service and catalog commerce management center customization

What this exercise is about .....	1
Lab requirements .....	1
What you should be able to do .....	1
Introduction .....	2
Exercise instructions .....	5
Part 1: Adding the new tables to the WebSphere Commerce schema .....	6
Part 2: Generating the physical data objects that represent the custom tables .....	8
Part 3: Configuring the catalog Web service to include the new data .....	10
Part 4: Create a custom Get Data tag configuration to request the new data in the Management Center. ....	14
Part 5: Extending the catalog resource bundle with translatable text.....	15
Part 6: Customizing the catalog management tool to display the new data .....	18
Part 7: Deploying and validating the customization. ....	21
Part 8: Solution instructions .....	25

### What this exercise is about

The objective of this lab is to provide you with an understanding of the customization touch points for the WebSphere Commerce programming model introduced in Feature Pack 3.01.

In this exercise you will add two new tables to the WebSphere Commerce schema and customize the catalog Web service to get and change the new data. You will then customize the catalog tool in the management center to display and update data in the new tables.

This lab is provided **AS-IS**, with no formal IBM support.

### Lab requirements

List of system and software requirements for the student to complete the lab.

- WebSphere Commerce Developer with WebSphere Application Server.
- WebSphere Commerce Feature Pack 3 with the management center feature enabled.

### What you should be able to do

At the end of this lab you should be able to:

- Customize WebSphere Commerce Web services to get and change new data.
- Customize the management center tools to display and update additional data.

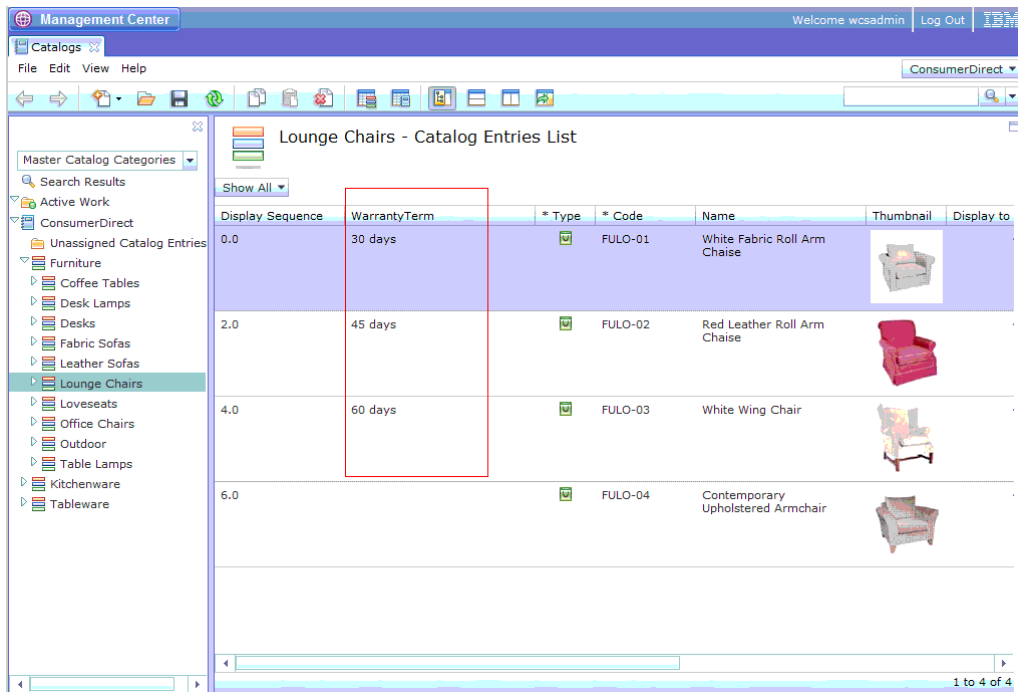
## Introduction

This exercise covers customizing the WebSphere Commerce Catalog Web Service and customizing the Catalog tool in the management center. This exercise is based upon adding warranty information and care instructions to catalog entries. A warranty consists of a term or number of days the warranty applies and a type, either limited or comprehensive. In this exercise, a care instruction applies to a specific catalog entry and language and consists of a text description.

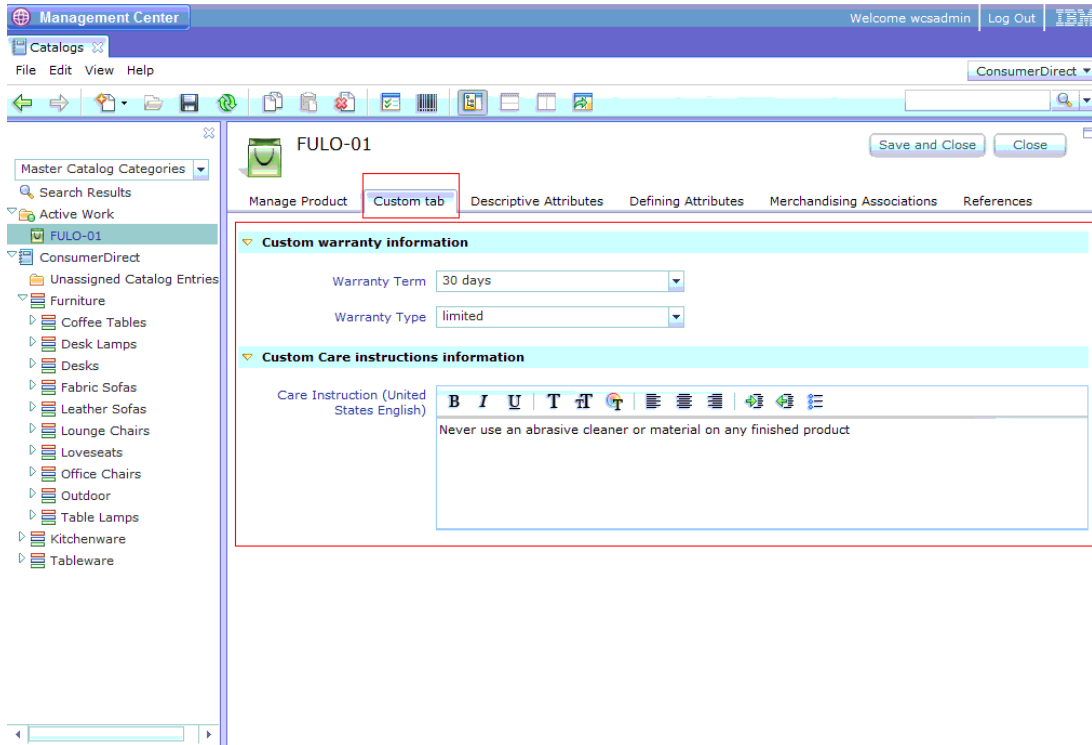
In the first few steps of this exercise you will extend the catalog Web service by adding new tables to the WebSphere Commerce schema and configuring the Web service to include the new data inside the user data element of the CatalogEntry noun. In the second half of this exercise you will customize the catalog tool to display and update the new data.

### Management center customization overview

Add a new column containing the warranty term to the “Catalog Entries List” view. The following image displays the updated view:



Add a new tab to the “Products Properties” view containing warranty and care instruction information. The following image displays the updated view:



### Catalog Web service customization overview

The WebSphere Commerce business object document (BOD) programming model separates processing into the business logic layer and the data service layer. The business logic layer performs business processes and enforces business rules for the logical business objects (or nouns) that it interacts with. The data service layer creates, reads, updates, or deletes business objects in the WebSphere Commerce database. This separation isolates the changes required to handle reading or managing new data added to business objects.

The CatalogEntry noun uses the UserData element as a data extension point to add new data without changing the logical model. In this exercise, warranty information will be added to the CatalogEntry noun UserData element to demonstrate the addition of language independent properties. This exercise defines warranty information per catalog entry.

The catalog description noun part supports the attributes element as a data extension point. Attributes provide the same functions as the UserData element. Product care instructions will be added to the CatalogEntryDescription noun part attributes element to show the addition of language dependent properties.

The following example CatalogEntry noun shows the customization of the CatalogEntry UserData element and CatalogEntryDescription attributes element:

```
<_cat:CatalogEntry catalogEntryTypeCode="ProductBean">
  <_cat:CatalogEntryIdentifier>
```

```
<_wcf:UniqueID>10251</_wcf:UniqueID>
  <_wcf:ExternalIdentifier ownerID="7000000000000000101">
    <_wcf:PartNumber>FULO-01</_wcf:PartNumber>
  </_wcf:ExternalIdentifier>
</_cat:CatalogEntryIdentifier>
<_cat:Description language="-1">
  <_cat:Name>White Fabric Roll Arm Chaise</_cat:Name>
  <_cat:Thumbnail>images/catalog/FULO_01_sm.jpg</_cat:Thumbnail>
  <_cat:FullImage>images/catalog/FULO_01.jpg</_cat:FullImage>
  <_cat:ShortDescription>Plumply padded for your ultimate
comfort.</_cat:ShortDescription>
  <_cat:Attributes name="careinstruction">Never use an abrasive
cleaner or material on any finished product.</_cat:Attributes>
</_cat:Description>
<_wcf:UserData>
  <_wcf:UserDataField name="warrantyTerm">30</_wcf:UserDataField>
  <_wcf:UserDataField
name="warrantyType">LIMITED</_wcf:UserDataField>
</_wcf:UserData>
</_cat:CatalogEntry>
```

---

## Exercise instructions

The directory locations are specified in the lab instructions using symbolic references, as follows:

Reference variable	Windows location
<WCDE_HOME>	The installation directory for WebSphere Commerce Developer.
<LAB_FILES>	The directory where you unpacked the Labfiles.zip auxiliary file archive.

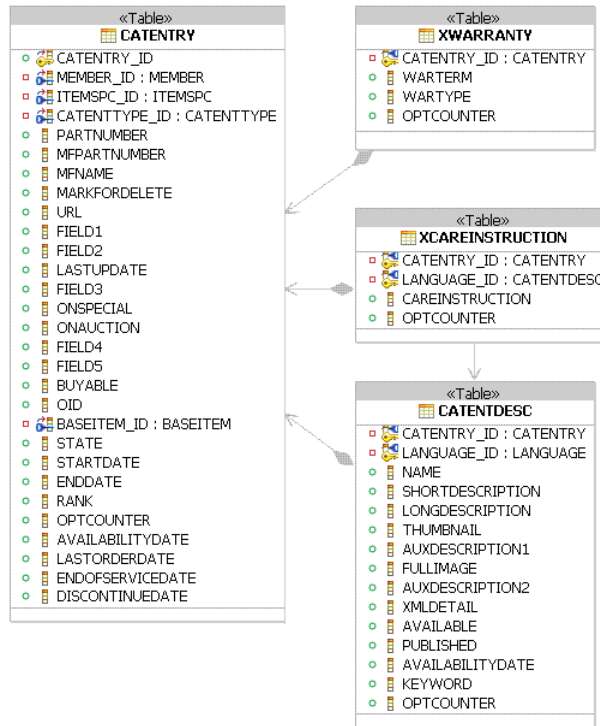
User names and passwords are specified in the lab instructions using symbolic references, as follows:

Reference variable	Windows location
<i>userid</i>	Your business user ID
<i>Password</i>	Your business user password

## Part 1: Adding the new tables to the WebSphere Commerce schema

In this step you will add new tables to contain warranty and care instruction information to the WebSphere Commerce schema.

The following diagram outlines the changes to the WebSphere Commerce schema:



The XWARRANTY table contains warranty information and the primary key is CATENTRY\_ID as warranty information is specific to a catalog entry. The XCAREINSTRUCTION table contains care instruction information and the primary key is CATENTRY\_ID and LANGUAGE\_ID as care instructions are specific for a catalog entry and for a language.

To link these tables into the WebSphere Commerce schema, the XWARRANTY table has a foreign key to the CATENTRY table and the XCAREINSTRUCTION table has a foreign key to the CATENTDESC table.

Additionally, the XCAREINSTRUCTION table has a foreign key to the CATENTRY table to provide support for the DSL search function. Refer to the information center for more information on the DSL search function.

To take advantage of the data service layer's user data (and attribute) support, the following restrictions apply when adding custom database tables:

- The custom tables must have a foreign key to the base table for the noun or noun part that will contain the data. A base table is a table that contains the unique key for the noun or noun part. For example, the CATENTRY table is the base table for the CatalogEntry noun. The CATENTDESC table is the base table for the CatalogEntryDescription noun part. These relationships are used when fetching or updating a noun or noun part to access the custom data.
- The relationship between the custom table and the base table must be a one-to-one relationship. User data is essentially a map of name-value pairs, where each name represents a database

column name and each value is the value in that database column. One-to-many relationships are not supported as each name in the map would collide.

- When adding a custom table for a noun part, a foreign key should also be created to the base table for the noun. This relationship provides support to the DSL search function and this relationship does not have to be one-to-one.

\_\_\_ 1. Create the custom tables and add sample data by performing the following steps:

\_\_\_ a. Open a command prompt.

\_\_\_ b. Run: `cd <WCDE_HOME>\bin`

\_\_\_ c. Run: `ij.bat "<LAB_FILES>\Server\SchemaCustomization.sql"`

---

The above command will create the two custom tables and populate these tables with sample data.

---

## Part 2: Generating the physical data objects that represent the custom tables

In this step, you will generate the physical data objects that represent the custom tables and configuration files for the Data Service Layer using the WebSphere Commerce Physical Service Data Object Generation wizard.

This wizard performs the following tasks:

- Creates an extension configuration folder for the Catalog service module if one does not already exist. The directory path is: `<WCDE_HOME>\xml\config\com.ibm.commerce.catalog-ext.`
- Generates a custom object-relational metadata file that describes the new custom tables and relationships. In this tutorial, the metadata file will describe the two new tables, XWARRANTY and XCAREINSTRUCTION, along with the three new relationships between tables:
  - XWARRANTY and CATENTRY
  - XCAREINSTRUCTION and CATENTRY
  - XCAREINSTRUCTION and CATENTDESC
  - This file is located at: `<WCDE_HOME>\xml\config\com.ibm.commerce.catalog-ext\wc-objectrelational-metadata.xml.`
- Generates a service data object (SDO) for each new and each modified table and places the data object inside the WebSphereCommerceServerExtensionsLogic project. A data object is generated for the following tables:
  - Each new custom table, XWARRANTY and XCAREINSTRUCTION.
  - Each modified WebSphere Commerce table, CATENTRY and CATENTDESC were modified by adding new relationships to the custom tables.
- Creates a utility Java class used by the Data Service Layer to find the custom data objects. This class is generated inside the WebSphereCommerceServerExtensionsLogic project.
- Creates an extension service module configuration file that provides the Data Service Layer with the location of the above utility class. This file is located at: `<WCDE_HOME>\xml\config\com.ibm.commerce.catalog-ext\wc-component.xml.`
- Creates an extension business object mediator configuration template to configure the Data Service Layer to add the new tables as user data elements to the CatalogEntry noun. This file is located at `<WCDE_HOME>\xml\config\com.ibm.commerce.catalog-ext\wc-business-object-mediator.xml.`

- \_\_\_ 1. Start WebSphere Commerce Developer. To start WebSphere Commerce Developer from your workstation desktop, double-click on the WebSphere Commerce Development Environment icon.
- \_\_\_ 2. Use the WebSphere Commerce Physical Service Data Object wizard to generate object-relational metadata and the physical service data objects that represent the custom tables.
  - \_\_\_ a. Ensure the WebSphere Commerce server is stopped.
  - \_\_\_ b. Select File > New > Other > WebSphere Commerce > Data Service Layer.



- \_\_\_ c. Click **Next**.
- \_\_\_ d. Select **Extend a default WebSphere Commerce service module**.
- \_\_\_ e. Click **Next**.
- \_\_\_ f. Enter the following information:
  - 1) Service module: com.ibm.commerce.catalog
  - 2) Extension class prefix: MyCompany
  - 3) Extension Java package name: com.mycompany.commerce.catalog

The screenshot shows a dialog box titled "Data Service Configuration" with a close button in the top right corner. The main heading is "Service Module Information". Below the heading is a descriptive text: "Allows customers to generate or regenerate data service configuration assets for a service module." There are three input fields: "Service module:" with a dropdown menu showing "com.ibm.commerce.catalog", "Extension class prefix:" with a text box containing "MyCompany", and "Extension Java package name:" with a text box containing "com.mycompany.commerce.catalog".

- \_\_\_ g. Click **Next**.
- \_\_\_ h. Select the **XWARRANTY** and **XCAREINSTRUCTION** tables.

The screenshot shows the same dialog box, but now on the "Table selection" tab. The heading is "Table selection" and the text says "Select the tables to include in the data service configuration." Below this is a table with a header row "Custom table name" and two rows of data. The first data row is "XCAREINSTRUCTION" with a checked checkbox. The second data row is "XWARRANTY" with a checked checkbox.

Custom table name	
<input checked="" type="checkbox"/> XCAREINSTRUCTION	
<input checked="" type="checkbox"/> XWARRANTY	

- \_\_\_ i. Click **Finish**.

## Part 3: Configuring the catalog Web service to include the new data

In this part, you will add or update the following configuration files for the catalog Web service.

- wc-query-MyCompanyCatalogEntry-get.tpl

This query template file contains the custom get queries used to select the CatalogEntry noun along with the new user data.

This file defines a new access profile, MyCompany\_All, which extends the IBM\_Details access profile and adds one new associated query, MyCompanyWarranty.

```
BEGIN_PROFILE
    name=MyCompany_All
    extends = IBM_Details

    BEGIN_ENTITY
        associated_sql_statement=MyCompanyGetCatalogEntryDetails
    END_ENTITY
END_PROFILE
```

This file also defines the MyCompanyGetCatalogEntryDetails associated query to get the new data.

```
BEGIN_ASSOCIATION_SQL_STATEMENT
    name=MyCompanyGetCatalogEntryDetails
    base_table=CATENTRY
    sql=
        SELECT
            CATENTRY.$COLS:CATENTRY$,
            CATENTDESC.$COLS:CATENTDESC$,
            XWARRANTY.$COLS:XWARRANTY$,
            XCAREINSTRUCTION.$COLS:XCAREINSTRUCTION$
        FROM
            CATENTRY
            LEFT OUTER JOIN XWARRANTY ON
            (CATENTRY.CATENTRY_ID = XWARRANTY.CATENTRY_ID)
            LEFT OUTER JOIN CATENTDESC ON
            (CATENTDESC.CATENTRY_ID = CATENTRY.CATENTRY_ID AND
            CATENTDESC.LANGUAGE_ID in ($CONTROL:LANGUAGES$))
            LEFT OUTER JOIN XCAREINSTRUCTION ON
            (CATENTRY.CATENTRY_ID = XCAREINSTRUCTION.CATENTRY_ID AND
            XCAREINSTRUCTION.LANGUAGE_ID in ($CONTROL:LANGUAGES$))
        WHERE
            CATENTRY.CATENTRY_ID IN ($ENTITY_PKS$) AND
            CATENTRY.MARKFORDELETE = 0
    END_ASSOCIATION_SQL_STATEMENT
```

- wc-query-MyCompanyCatalogEntry-update.tpl

This query template file contains the custom update queries used to update the CatalogEntry noun along with the new user data. To create these queries, the existing WebSphere Commerce CatalogEntry update queries are placed in a custom query template file and updated to include the new tables.

The WebSphere Commerce update queries for a CatalogEntry are found under <WCDE\_HOME>\xml\config\com.ibm.commerce.catalog\wc-query-CatalogEntry-update.tpl.

The following query template is used to update a CatalogEntryDescription and care instruction information (custom code is in bold font).

```
<!-- Custom query to be used by the ChangeCatalogEntryDescriptionMediator -->
BEGIN_XPATH_TO_SQL_STATEMENT
  name=/CatalogEntry[CatalogEntryIdentifier[(UniqueID=)]]/Description+MyCompany_CatalogEntryDescription_Update
  base_table=CATENTDESC
  sql=
    SELECT
      CATENTDESC.$COLS:CATENTDESC$,
      XCAREINSTRUCTION.$COLS:XCAREINSTRUCTION$
    FROM
      CATENTDESC
      JOIN XCAREINSTRUCTION ON (CATENTDESC.CATENTRY_ID =
      XCAREINSTRUCTION.CATENTRY_ID AND XCAREINSTRUCTION.LANGUAGE_ID =
      CATENTDESC.LANGUAGE_ID)
    WHERE
      CATENTDESC.CATENTRY_ID IN (?UniqueID?)
```

---

This query also defined a new access profile, MyCompany\_CatalogEntryDescription\_Update, to distinguish this query from the original query provided by WebSphere Commerce.

---

- wc-business-object-mediator.xml

This file instructs the CatalogEntry Change mediators to use the custom update access profiles defined in the wc-query-MyCompanyCatalogEntry-update.tpl file and to include the contents of the new tables as user data in the CatalogEntry noun and CatalogEntryDescription noun part.

- MyCompanyCatalogAccessControlPolicies.xml

In the preceding steps, you defined MyCompany\_All, a new access profile for get requests. By default, only users with the site administrator roles will have access to this data. This access control policy will add the MyCompany\_All access profile to the CatalogEntry all users group.

\_\_\_ 1. Add the predefined query template files to get and update the custom data.

\_\_\_ a. Open a Windows Explorer session.

\_\_\_ b. Navigate to:

<LAB\_FILES>\FoundationExercise\Server

\_\_\_ c. Select the following two files:

- wc-query-MyCompanyCatalogEntry-get.tpl
- wc-query-MyCompanyCatalogEntry-update.tpl

\_\_\_ d. Copy the above files to:

<WCDE\_installdir>\xml\config\com.ibm.commerce.catalog-ext\

\_\_\_ 2. Update the generated business object mediator configuration template for this customization.

\_\_\_ a. Open a Windows Explorer session.

\_\_\_ b. Navigate to <WCDE\_installdir>\xml\config\com.ibm.commerce.catalog-ext\

- \_\_ c. Edit the wc-business-object-mediator.xml file and examine this section, near the bottom of the file, which maps data from the XWARRANTY table to the CatalogEntry noun. Keep the file open for a subsequent step.

```
<_config:child-property-mapping
relationshipName="XwarrantyForCatentry">
  <_config:userDataProperty logicalPropertyName="catentry_id"
  physicalPropertyName="catentry_id" />
  <_config:userDataProperty logicalPropertyName="warterm"
  physicalPropertyName="warterm" />
  <_config:userDataProperty logicalPropertyName="wartype"
  physicalPropertyName="wartype" />
  <_config:userDataProperty logicalPropertyName="optcounter"
  physicalPropertyName="optcounter" />
</_config:child-property-mapping>
```

By default, the WebSphere Commerce physical data object wizard adds all columns of new tables to the user data configuration. In the above snippet, the entries for the following columns may be removed: catentry\_id, optcounter.

- \_\_ d. Open a Windows Explorer session.

- \_\_ e. Edit the file:

```
<LABFILES>\FoundationExercise\bomSnippet.txt
```

- \_\_ f. Copy and paste the code in <LabExercise>\Server\bomSnippet.txt underneath the first "<\_config:object" element in the wc-business-object-mediator.xml file.

```
<_config:object
logicalType="com.ibm.commerce.catalog.facade.datatypes.CatalogEntryType"
physicalType="com.ibm.commerce.catalog.facade.server.entity.datatypes.CatalogEntry">
  <_config:mediator
  interfaceName="com.ibm.commerce.foundation.server.services.dataaccess.
  bom.mediator.ChangeBusinessObjectMediator"
  className="com.ibm.commerce.catalog.facade.server.services.dataaccess.
  bom.mediator.ChangeCatalogEntryMediator"
  updateAccessProfile="MyCompany_CatalogEntry_Update">
```

- \_\_ g. Save the wc-business-object-mediator.xml file.

This step updates the default access profile used by the Change CatalogEntry mediators to the custom update access profiles defined in the wc-query-MyCompanyCatalogEntry-update.tpl file in a previous step.

```
<_config:part-mediator-implementation
className="com.ibm.commerce.catalog.facade.server.services.dataaccess
.bom.mediator.ChangeCatalogEntryDescriptionMediator"
updateAccessProfile="MyCompany_CatalogEntryDescription_Update" />
```

- \_\_\_ 3. Create and load an access control policy for the MyCompany\_All access profile.

This policy defines an action for the MyCompanyAll access profile.

```
<Action Name="GetCatalogEntry.MyCompanyAll"
CommandName="GetCatalogEntry.MyCompany_All" />
```

And adds the new action to the existing Catalog-CatalogEntry-AllUsers\_AccessProfileActionGroup

```
<ActionGroup Name="Catalog-CatalogEntry-AllUsers-
AccessProfileActionGroup" OwnerID="RootOrganization">
  <ActionGroupAction Name="GetCatalogEntry.MyCompanyAll" />
</ActionGroup>
```

---

Access control policies for update access profiles are not needed as they are used by change, process, and sync commands. If a user has the authority to run these commands, they will have implicit authority to use these access profiles. In this exercise, users with the authority to change a CatalogEntry noun are able to create, update, and delete the new data.

---

\_\_ a. Open a Windows Explorer session.

\_\_ b. Navigate to:

```
<Labfiles>\FoundationExercise\Server\MyCompanyCatalogAccessControlPolicies.xml
```

\_\_ c. Select following file:

- MyCompanyCatalogAccessControlPolicies.xml

\_\_ d. Copy the above file to:

```
<WCDE_HOME>\xml\policies\xml
```

\_\_ e. Open a command prompt.

\_\_ f. Run: `cd <WCDE_HOME>\bin`

\_\_ g. Run: `acpload`

```
"<WCDE_HOME>\xml\policies\xml\MyCompanyCatalogAccessControlPolicies.xml"
```

**You should see this output**

```
C:\WCToolkitEE60\bin>acpload
```

```
".\xml\policies\xml\MyCompanyCatalogAccessControlPolicies.xml"
```

```
Running XMLTransform...
```

```
Running Id Resolver...
```

```
Running MassLoader...
```

```
Examine acpload.log to ensure that everything completed successfully.
```

---

This access control policy updates the existing catalog access control policy to allow all users to get and display the MyCompany\_All access profile.

---

\_\_\_ 4. Examine the log in `<WCDE_HOME>\logs\acpload.log` and verify that there are no errors. Your log should look something like this:

```
Running XMLTransform...
```

```
Running Id Resolver...
```

```
Running MassLoader...
```

```
JDBCW->Inside clearMap
```

## Part 4: Create a custom Get Data tag configuration to request the new data in the management center.

The Get Data tag is used to simplify the creating of a Get request in the WebSphere Commerce BOD programming model. The Get Data tag configuration defined in this step will be used by the management center to request data with the MyCompany\_All access profile defined in Part 3.

- \_\_\_ 1. Create the custom Get Data tag configuration by performing the following steps:
  - \_\_\_ a. In WebSphere Commerce Developer, navigate to the following in the Project Explorer view.  
Dynamic Web Projects\LOBTools\WebContent\WEB-INF\config
  - \_\_\_ b. Right click on the **config** folder
  - \_\_\_ c. Select New Folder.
  - \_\_\_ d. In the Folder name field, enter com.ibm.commerce.catalog-ext.
  - \_\_\_ e. Click Finish.
  - \_\_\_ f. Right click on the com.ibm.commerce.catalog-ext folder.
  - \_\_\_ g. Select New > Other > Simple > File.
  - \_\_\_ h. Click Next.
  - \_\_\_ i. In the File name field, enter get-data-config.xml.
  - \_\_\_ j. Click Finish.
  - \_\_\_ k. Copy and paste the following configuration into the get-data-config.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<wcf:get-data-config
xmlns:wcf="http://www.ibm.com/xmlns/prod/commerce/foundation"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/xmlns/prod/commerce/foundation
../xsd/get-data-config.xsd ">
  <expression-builder>
    <name>getCatalogEntryDetailsByParentCatalogGroupId</name>
    <data-type-name>CatalogEntry</data-type-name>
    <param>
      <name>accessProfile</name>
      <value>MyCompany_All</value>
    </param>
  </expression-builder>
</wcf:get-data-config>
```

**Note:** You can also find this text in the file under the solutions directory starting at <Labfiles>\FoundationExercise\Solution.

The above configuration will override the existing configuration for the **getCatalogEntryDetailsByParentCatalogGroupId** expression to use the MyCompany\_All access profile.

- \_\_\_ l. Save the file

---

## Part 5: Extending the catalog resource bundle with translatable text

---

The following properties file contains labels, names, and values for the code added in the previous part.

---

- \_\_\_ 1. Add the provided properties file containing translatable text.
  - \_\_\_ a. In WebSphere Commerce Developer, navigate to the following in the Project Explorer view.  
`Dynamic Web Projects\LOBTools\Java Resources\src`
  - \_\_\_ b. Right click on the **src** folder.
  - \_\_\_ c. Select New > Package.
  - \_\_\_ d. In the Name field, enter **extension.catalog**.
  - \_\_\_ e. Click Finish.
  - \_\_\_ f. Open a Windows Explorer session.
  - \_\_\_ g. Navigate to:  
`<Labfiles>\FoundationExercise\Client`
  - \_\_\_ h. Select following file:
    - `CatalogLOB_en_US.properties`
  - \_\_\_ i. Copy the above file to:  
`<WCDE_HOME>\workspace\LOBTools\src\extension\catalog`
- \_\_\_ 2. Extend the Catalog resource bundle to include the custom property file.
  - \_\_\_ a. In WebSphere Commerce Developer, navigate to the following in the Project Explorer view.  
`Dynamic Web Projects\LOBTools\WebContent\WEB-INF\src\lzx`
  - \_\_\_ b. Right click on the **lzx** folder.
  - \_\_\_ c. Select New > Folder
  - \_\_\_ d. In the folder name field, enter mycompany. Click Finish.
  - \_\_\_ e. Right click on the mycompany folder.
  - \_\_\_ f. Select New > Folder.
  - \_\_\_ g. In the folder name field, enter catalog. Click Finish.
  - \_\_\_ h. Open a Windows Explorer session.
  - \_\_\_ i. Navigate to:  
`<LabExercise>\Client`
  - \_\_\_ j. Select following file:
    - `extCatalogManagementResourceBundle.lzx`
  - \_\_\_ k. Copy the above file to:  
`<WCDE_HOME>\workspace\LOBTools\WebContent\WEB-INF\src\lzx\mycompany\catalog\extCatalogManagementResourceBundle.lzx`

- \_\_\_ 3. Register the extcatalogManagementResourceBundle.lzx file..
  - \_\_\_ a. In WebSphere Commerce Developer, navigate to the following in the Project Explorer view.  
Dynamic Web Projects\LOBTools\WebContent\WEB-INF\src\lzx\commerce\catalog
  - \_\_\_ b. Open **CatalogExtensionsLibrary.lzx**.
  - \_\_\_ c. Update this file to include the customized resource bundle by adding the following configuration:

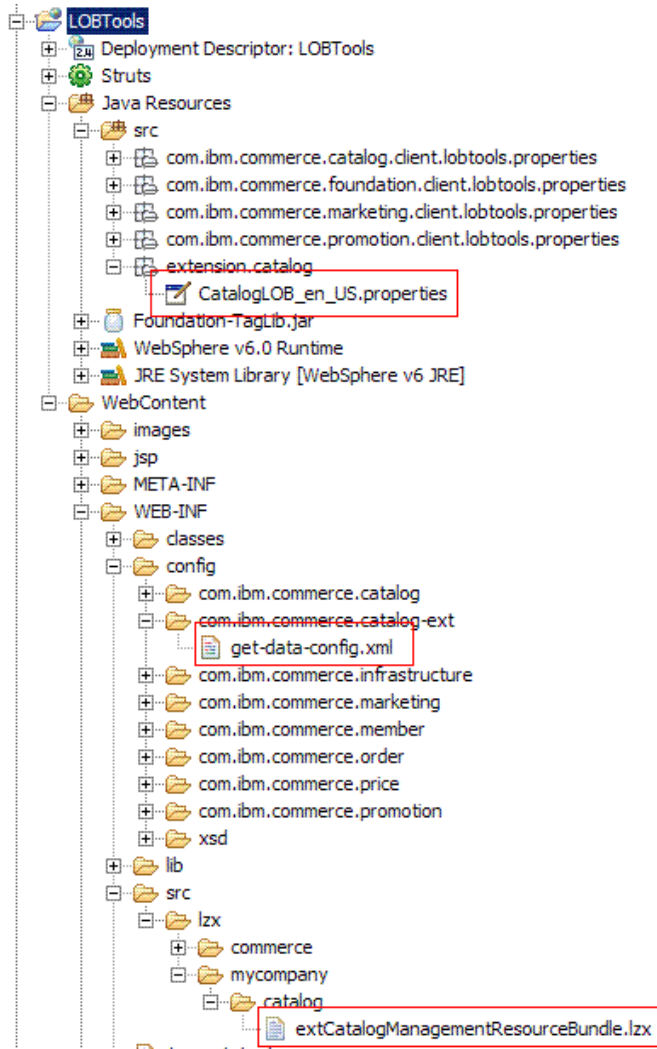
```
<library>
  <!-- File to add customer libraries -->
  <include
    href="..\..\mycompany\catalog\extCatalogManagementResourceBundle.lzx"
  />
</library>
```
  - \_\_\_ d. Save the file.
- \_\_\_ 4. Refresh the LOBTools Web project to pick up the new files.
  - \_\_\_ a. In WebSphere Commerce Developer, right click on the LOBTools project.
  - \_\_\_ b. Select Refresh.

---

The following screen capture highlights the files added to the client so far:

---





## Part 6: Customizing the catalog management tool to display the new data

This exercise defines a warranty term as 30, 45, or 60 days and a warranty type as either limited or comprehensive. To ensure these specific values are used, you will define combo boxes that support only these values.

### 1. Define combo boxes for the WarrantyTerm and WarrantyType properties.

#### a. In WebSphere Commerce Developer navigate to the following in the Project Explorer view

```
Dynamic Web Projects\LOBTools\WebContent\WEB-INF\src\lzx\commerce\catalog\objectDefinitions\ProductPrimaryObjectDefinition.lzx
```

#### b. Open ProductPrimaryObjectDefinition.lzx.

#### c. Search on name="catProductPrimaryObjectDefinition"

#### d. Inside the class definition for "catProductPrimaryObjectDefinition", define the combo boxes for the warranty term and warranty type. Insert the following code:

```
<wcfPropertyDefinition propertyName="x_warterm" type="number">
  <wcfPropertyValue
    displayName="{extCatalogResources.productWarranty_DisplayNameForTerm
1.string}" value="30"/>
  <wcfPropertyValue
    displayName="{extCatalogResources.productWarranty_DisplayNameForTerm
2.string}" value="45"/>
  <wcfPropertyValue
    displayName="{extCatalogResources.productWarranty_DisplayNameForTerm
3.string}" value="60"/>
</wcfPropertyDefinition>

<wcfPropertyDefinition propertyName="x_wartype" type="string">
  <wcfPropertyValue
    displayName="{extCatalogResources.productWarranty_DisplayNameForType
1.string}" value="LIMITED"/>
  <wcfPropertyValue
    displayName="{extCatalogResources.productWarranty_DisplayNameForType
2.string}" value="COMPREHENSIVE"/>
</wcfPropertyDefinition>
```

before these lines at the bottom of the file:

```
</class>
</library>
```

### 2. Add the WarrantyTerm combo box to the "Catalog Entries List" view.

#### a. In WebSphere Commerce Developer, open CatalogEntryGrid.lzx.

```
Dynamic Web Projects\LOBTools\WebContent\WEB-INF\src\lzx\commerce\catalog\listViewDefinitions\CatalogEntryGrid.lzx
```

#### b. Search on name="catCatalogEntryBrowseGrid"

#### c. Find the following element:

```
<wcfGridText editable="false" name="catentryId"
  objectPath="CatalogEntry" propertyName="catentryId"
  text="{catalogResources.productUniqueId_ColumnHeader.string}"
  visible="false" width="90"/>
```

#### d. Insert the following after the above element.

```

        <wcfGridComboBox name="WarrantyTerm" editable="true"
            objectPath="CatalogEntry"
            propertyName="x_warterm"
            text="{extCatalogResources.productWarranty_ColumnHeader.string}"
            width="90" visible="true"/>

```

\_\_\_ 3. Create the new tab to display all warranty and care instruction information.

\_\_\_ a. In WebSphere Commerce Developer, open **ProductPropertiesView.lzx**.

```

Dynamic Web Projects\LOBTools\WebContent\WEB-
INF\src\lzx\commerce\catalog\propertiesView\ProductPropertiesView.lzx

```

\_\_\_ b. Find the following element:

```

<!-- Tab: Manage Product. This tab contains general information about
the selected product such as name, and description. -->
    <wcfPropertyTabPane name="manageProductTab"
text="{catalogResources.manageProductTab.string}">
        <!-- Property Pane: Manage Product. This is an
instantiation of the property pane class which contains general
product details. -->
            <catManageProduct/>
    </wcfPropertyTabPane>

```

\_\_\_ c. Add the following code after the above element to define the custom tab.

```

<!-- Tab: CustomTab. This tab contains warranty information and care
instructions for a product. -->
    <wcfPropertyTabPane name="productCustomTab"
text="{extCatalogResources.customTabHeader.string}">
        <!-- Property Pane: Custom data. This is an instantiation of
the property pane class which contains custom warranty and care
instruction data. -->
            <catCustomPane/>
    </wcfPropertyTabPane>

```

\_\_\_ 4. Define the contents of the new tab.

\_\_\_ a. In WebSphere Commerce Developer, open **CatalogPropertyPane.lzx**

```

Dynamic Web Projects\LOBTools\WebContent\WEB-
INF\src\lzx\commerce\catalog\propertiesView\CatalogPropertyPane.lzx

```

\_\_\_ b. Find the following element:

```

<class extends="wcfPropertyPane" name="catManageKit">
    <catManageGeneralKitInformation/>
    <catManagePublishingInformation/>
    <catManageDisplayInformation/>
    <catManagePricingInformation/>
    <!-- This tag is disabled by default. To enable it, uncomment
the tag below -->
    <!-- <catManageAdditionalInformation/> -->
</class>

```

\_\_\_ c. Add the following code after the above element to define a warranty group consisting of the two warranty combo boxes and a care instruction group consisting of a rich text editor displaying the care instruction.:

```

<!--
@Class: catCustomPane
@Extends: wcfPropertyPane
@Purpose: MyCompany custom tab: The page that displays warranty and
care instruction information.
-->
<class name="catCustomPane" extends="wcfPropertyPane">
    <wcfPropertyGroup open="true" groupTitle="Warranty Information">

```

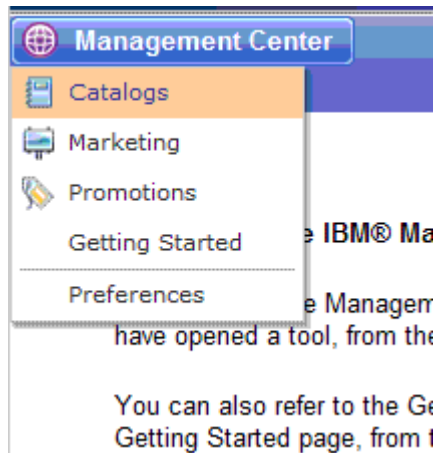
```
<wcfPropertyCombobox promptText="WarrantyTerm" width="200"
editable="true" propertyName="x_warterm" />
<wcfPropertyCombobox promptText="WarrantyType" width="200"
editable="true" propertyName="x_wartype" />
</wcfPropertyGroup>

<wcfPropertyGroup open="true" groupTitle="Care Instruction">
  <wcfPropertyRichTextEditor
    objectPath="CatalogEntryDescription"
    propertyName="xdesc_careinstruction"
    promptText="CareInstruction" />
</wcfPropertyGroup>
</class>
```

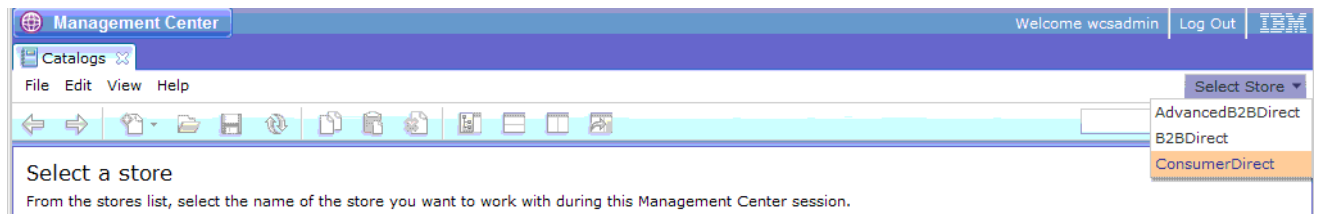
- \_\_\_\_ 5. Save all open editors with Shift-Ctrl-S and wait for the workspace build to complete.

## Part 7: Deploying and validating the customization.

- \_\_\_ 1. Publish the customization to the WebSphere Commerce test server.
  - \_\_\_ a. In WebSphere Commerce Developer Servers view, right click on WebSphere Commerce Test Server and select Start.
  - \_\_\_ b. The server takes approximately two minutes to start. Once the server status changes to Started, re-publish the WebSphere Commerce project to the WebSphere Commerce Test Server: On the **Server** page, right-click the WebSphere Commerce Test Server and select **Publish**.
- \_\_\_ 2. Test the customization in the management center (CMC).
  - \_\_\_ a. Open Internet Explorer and enter URL: <https://localhost:8000/lobtools>.
  - \_\_\_ b. Log on to CMC by using the user ID and password specified at the beginning of this exercise.
  - \_\_\_ c. Select the Catalog tool under the top left menu of the management center.







- \_\_\_ d. Select the Consumer Direct store.



**NOTE:** The sample data added in step one of this exercise is for catalog entries belonging to the Consumer Direct store.

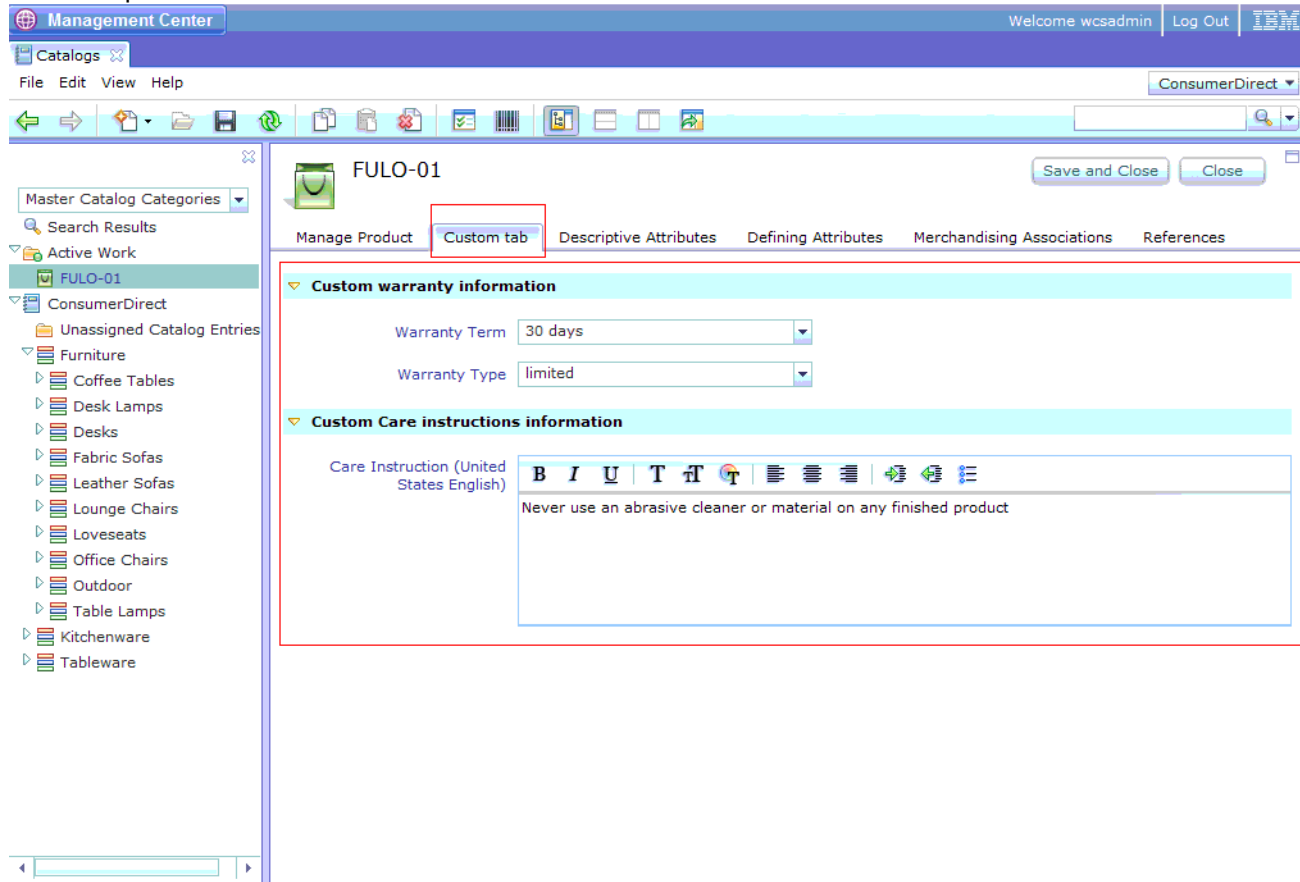
- \_\_\_ e. In the Master Catalog Categories panel, navigate to ConsumerDirect > Furniture.
- \_\_\_ f. Click on Lounge Chairs.
- \_\_\_ g. Verify your results with the following screen capture:

The screenshot shows the Management Center interface with a table titled "Lounge Chairs - Catalog Entries List". The table has the following columns: Display Sequence, WarrantyTerm, \* Type, \* Code, Name, Thumbnail, and Display to. The first row is highlighted in blue and contains the following data: 0.0, 30 days, [checked], FULO-01, White Fabric Roll Arm Chaise, [Thumbnail], and [Display to]. The second row contains: 2.0, 45 days, [checked], FULO-02, Red Leather Roll Arm Chaise, [Thumbnail], and [Display to]. The third row contains: 4.0, 60 days, [checked], FULO-03, White Wing Chair, [Thumbnail], and [Display to]. The fourth row contains: 6.0, [checked], FULO-04, Contemporary Upholstered Armchair, [Thumbnail], and [Display to]. A red box highlights the "WarrantyTerm" column for the first three rows. The interface includes a navigation pane on the left with "Lounge Chairs" selected, and a status bar at the bottom right showing "1 to 4 of 4".

Display Sequence	WarrantyTerm	* Type	* Code	Name	Thumbnail	Display to
0.0	30 days	<input checked="" type="checkbox"/>	FULO-01	White Fabric Roll Arm Chaise		
2.0	45 days	<input checked="" type="checkbox"/>	FULO-02	Red Leather Roll Arm Chaise		
4.0	60 days	<input checked="" type="checkbox"/>	FULO-03	White Wing Chair		
6.0		<input checked="" type="checkbox"/>	FULO-04	Contemporary Upholstered Armchair		

\_\_ h. Double click on the White Fabric Roll Arm Chair to open the property panel view.

- \_\_\_ i. Click on the Custom tab and verify the warranty and care instruction information with the following screen capture:



- \_\_\_ j. Click on the Warranty Term combo box and select 60 days.
- \_\_\_ k. Click on the "Save and Close" button.
- \_\_\_ l. Verify your results with the following screen capture:

The screenshot displays the IBM Management Center interface. The main window title is "Lounge Chairs - Catalog Entries List". On the left, a navigation tree shows "Master Catalog Categories" with "Lounge Chairs" selected. The main area contains a table with the following data:

Display Sequence	WarrantyTerm	* Type	* Code	Name	Thumbnail	Display to
0.0	60 days	<input checked="" type="checkbox"/>	FULO-01	White Fabric Roll Arm Chaise		
2.0	45 days	<input checked="" type="checkbox"/>	FULO-02	Red Leather Roll Arm Chaise		
4.0	60 days	<input checked="" type="checkbox"/>	FULO-03	White Wing Chair		
6.0		<input checked="" type="checkbox"/>	FULO-04	Contemporary Upholstered Armchair		

At the bottom of the window, a status bar displays the message: "Save action completed for Product FULO-01." The bottom right corner of the window shows "1 to 4 of 4".



## Part 8: Solution instructions

You can speed up your solution of the exercise by finding the solution files in <Labfiles>\FoundationExercise\Solution. You will see these directories which are the results of the steps listed in the Purpose column. You can use the contents of the files to help you edit the originals. You can also use the files to replace the originals to save you editing the files.

Project directory	Purpose
WebSphereCommerceServerExtensionsLogic	Output of steps in Part 2
LOBTools	Management Center project modified in Parts 4-6
com.ibm.commerce.catalog-ext	Contents of directory <WCDE_installdir>\xml\config\com.ibm.commerce.catalog-ext created in Part 3