

IBM WebSphere Commerce



Customizing Product Management tools in WebSphere Commerce Accelerator

Version 5.5

IBM WebSphere Commerce



Customizing Product Management tools in WebSphere Commerce Accelerator

Version 5.5

Note:

Before using this information and the product it supports, be sure to read the general information in the Notices section.

First Edition, April 2004

This edition applies to Version 5.5 of IBM WebSphere Commerce Business Edition, IBM WebSphere Commerce Professional Edition, IBM WebSphere Commerce — Express Edition, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality.

IBM welcomes your comments. You can send your comments by using the online IBM WebSphere Commerce documentation feedback form, available at the following URL:

www.ibm.com/software/webservers/commerce/rcf.html

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2004. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this document	v
Document description	v
Knowledge requirements	v
Conventions used in this document	v
Path variables.	vi
Updates to this document.	vi

Chapter 1. Customizing catalogs in the WebSphere Commerce Accelerator 1

Chapter 2. Product Management tools search	3
Architecture	3
Customization scenario	3
Searching by keyword	3
Assets	10

JSP assets	10
XML assets	10
Database assets	11

Chapter 3. Product Management tools tabs and columns 13

Architecture	13
Customization scenarios	13
Customizing tabs and columns	13
Supporting customized product information	17
Assets	26
JSP assets	26
XML assets	27
Database assets	28

Notices 29

About this document

Document description

This document serves as an overview of the concepts involved in customizing the WebSphere® Commerce Accelerator Product Management tools for catalogs. It addresses the high level architecture of how the business users interact with the user interface. Supplementary documents, released as they become available, build upon the knowledge developed in the *WebSphere Commerce Accelerator Customization Guide*, and provide the detailed information required to customize the particular components of the WebSphere Commerce Accelerator. These supplementary documents also act as a resource, listing the components and assets upon which the various components depend.

Knowledge requirements

To customize the WebSphere Commerce Accelerator, you require knowledge of the following:

- HTML and XML
- Structured Query Language (SQL)
- Java™ programming
- JavaServer Pages (JSP) technology
- IBM® WebSphere Commerce Studio or WebSphere Commerce — Express Developer Edition

Refer to the *WebSphere Commerce Programming Guide and Reference* for more information on customizing WebSphere Commerce. This book is available from the Technical Library Web site (www.ibm.com/software/commerce/library/).

Conventions used in this document

This document uses the following highlighting conventions:

Boldface type indicates commands or graphical user interface (GUI) controls such as names of fields, buttons, or menu choices.

Monospaced type indicates examples of text you enter exactly as shown, as well as directory paths.

Italic type is used for emphasis and variables for which you substitute your own values.



This icon marks a tip — additional information that can help you complete a task.

Important

These sections highlight especially important information.

Attention





These sections highlight information intended to protect your data.

Path variables

This document uses the following variables to represent directory paths:

WC_installdir

This is the installation directory for WebSphere Commerce. The following are the default installation directories for WebSphere Commerce on various operating systems:

-  /usr/WebSphere/CommerceServer55/
-  /QIBM/ProdData/CommerceServer55/
-   /opt/WebSphere/CommerceServer55/
-  C:\Program Files\WebSphere\CommerceServer55\

WAS_installdir

This is the installation directory for WebSphere Application Server. The following are the default installation directories for WebSphere Application Server on various operating systems:

-  /usr/WebSphere/AppServer/
-  /QIBM/ProdData/WebA55/
-   /opt/WebSphere/AppServer/
-  C:\Program Files\WebSphere\AppServer\

WCDE_installdir

This is the installation directory for the WebSphere Commerce - Express development environment. The default installation directory for the WebSphere Commerce - Express development environment is C:\WebSphere\CommerceDev55.

Updates to this document

The latest version of this document is available as a PDF file from the Technical Library Web site.

Chapter 1. Customizing catalogs in the WebSphere Commerce Accelerator

This document discusses how to customize the Product Management tools user interface in the WebSphere Commerce Accelerator. These tasks provide additional catalog functionality. This document should only be read after you have completed reading the primary *WebSphere Commerce Accelerator Customization Guide*, as it builds upon information and concepts contained in that document.

This document provides customization details for the following components in the Product Management tools:

- using keyword search capabilities for catalog entries,
- creating view tabs and columns,
- customized product information.

Chapter 2. Product Management tools search

Architecture

The Catalog Entry Search page provides an entry point to the catalog tools available in the WebSphere Commerce Accelerator, including the Product Management tools dynamic table, and the user interfaces for merchandising associations, and bundles and kits.

The Catalog Entry Search page collects search criteria from a user and forwards the data to the corresponding tooling page. The page will activate the search data bean using the given data and a set of catalog entry IDs will be returned. The tooling page will then populate the CatalogEntryDataBean or the CatalogEntryAccessBean using these catalog entries, which are on the results page.

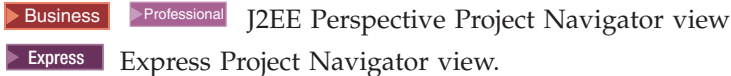
The search data bean supports searching on keywords but the JSP file does not use this feature. An additional search field may be useful to customize search results.

Customization scenario

You can customize the Catalog Entry Search page by supporting a keyword search field. You will need to modify two JSP files — CatEntrySearchDialog.jsp and CatalogSearchUtil.jsp — which use the AdvancedCatEntrySearchListDataBean.

Searching by keyword

This scenario customizes the Catalog Entry Search page by supporting keyword search, providing a new level of search results.

1. To protect your customized data from being overwritten during migration to a future version, or during installation of a future fix pack, or some similar event, it must be created in a safe place, separate from the WebSphere Commerce assets. This procedure creates a custom version of the CatEntrySearchDialog.jsp file within the WebSphere Commerce development environment:
 - a. Open the WebSphere Commerce development environment (**Start > Programs > IBM WebSphere Commerce development environment**) and switch to the
 J2EE Perspective Project Navigator view
Express Project Navigator view.
 - b. Navigate to the CommerceAccelerator project.
 - c. Navigate to the Web Content/tools folder.
 - d. Right-click on the folder, and select **New**, and then **Folder**. In the **Folder name** field, type custom.
 - e. Navigate to the Web Content/tools/catalog folder.
 - f. Right-click on the CatEntrySearchDialog.jsp file, and select **Copy**.
 - g. Right-click on the Web Content/tools/custom folder, and select **Paste**.
 - h. Navigate to the Web Content/tools/custom folder.
 - i. Double-click the new file to open it in a text editor.
 - j. Create the input box in the Catalog Entry Search page. Locate the following code section:

```

<TR>
  <TD COLSPAN="3"><%=categoryFindNLS.get("catalogSearch_name"%>
</TR>
<TR>
  <TD><INPUT TYPE="text" NAME="name" CLASS="input" MAXLENGTH="64"></TD>
  <TD></TD>
  <TD>
    <SELECT NAME="nameOp" CLASS="input">
      <OPTION VALUE="LIKE"><%=categoryFindNLS.get("catalogSearch_operator_like"%>
      </OPTION>
      <OPTION VALUE="EQUAL"><%=categoryFindNLS.get("catalogSearch_operator_exact"%>
      </OPTION>
    </SELECT>
  </TD>
</TR>
<TR>
  <TD COLSPAN=3><INPUT TYPE="checkbox">
  NAME="searchScope"><%=categoryFindNLS.get("catalogSearch_include_desc"%>
</TR>
<TR><TD></TD></TR>

```

Immediately under this section, add the following code lines:

```

<TR>
  <TD COLSPAN="3">
  <%=categoryFindNLS.get("categoryFindKeyword"%>
</TR>
<TR>
  <TD><INPUT TYPE="text" NAME="keyword" CLASS="input" MAXLENGTH="64"></TD>
  <TD></TD>
  <TD>
    <SELECT NAME="keywordOp" CLASS="input">
      <OPTION VALUE="LIKE"><%=categoryFindNLS.get("catalogSearch_operator_like"%>
      </OPTION>
      <OPTION VALUE="EQUAL"><%=categoryFindNLS.get("catalogSearch_operator_exact"%>
      </OPTION>
    </SELECT>
  </TD>
</TR>
<TR><TD></TD></TR>

```

- k. Store the data relating to the keyword into the same object that stores all the parameters. Locate the following code section in the `findAction()` function:

```

urlPara.skuOp      = document.all.skuOp.value;
urlPara.name      = document.all.name.value;
urlPara.nameOp    = document.all.nameOp.value;

```

Immediately after this section, add the following code lines:

```

urlPara.keyword    = document.all.keyword.value;
urlPara.keywordOp  = document.all.keywordOp.value;

```

- l. Save the file, but do not close the development environment.
2. Register the `CatEntrySearchDialog.jsp` file into the `VIEWREG` table with the following SQL statement:

```

update viewreg set PROPERTIES = 'docname=tools/custom/CatEntrySearchDialog.jsp'
where VIEWNAME = 'CatEntrySearchDialog'

```
3. This procedure creates a custom version of the `CatalogSearchUtil.jsp` file within the WebSphere Commerce development environment:
 - a. In the WebSphere Commerce development environment, navigate to the `CommerceAccelerator/Web Content/tools/catalog` folder.
 - b. Right-click on the `CatalogSearchUtil.jsp` file, and select **Copy**.
 - c. Right-click on the `Web Content/tools/custom` folder and select **Paste**.
 - d. Navigate to the `Web Content/tools/custom` folder.
 - e. Double-click the new file to open it in a text editor.

- f. Accept the keyword as a parameter to be passed into the search. Locate the following code in the `setCatEntryParameters()` function:

```
public void setCatEntryParameters(JSPHelper helper, AdvancedCatEntrySearchListDataBean
catEntrySearchDB) {
    String catalogID, name, nameOp, searchScope, displayNum, sortBy, storeId,
        languageId;
    String startCategory, startCategoryOp, sku, skuOp, manuNum, manuNumOp, manuName,
        manuNameOp, published, notPublished;
    String searchProduct, searchItem, searchPackage, searchBundle, searchDynamicKit,
        startIndex;
```

Immediately under this section, add the following line:

```
String keyword, keywordOp;
```

- g. Obtain the value of the keyword from the input. Locate the following code in the `setCatEntryParameters()` function:

```
storeId          = helper.getParameter("storeId");
languageId       = helper.getParameter("languageId");
name             = helper.getParameter("name");
nameOp          = helper.getParameter("nameOp");
```

Immediately under this section, add the following code lines:

```
keyword          = helper.getParameter("keyword");
keywordOp       = helper.getParameter("keywordOp");
```

- h. Trim the keyword input. Locate the following code in the `setCatEntryParameters()` function:

```
if (sku != null) {
    sku = sku.trim();
}
```

Immediately under this section, add the following code lines:

```
if (keyword != null) {
    keyword = keyword.trim();
}
```

- i. Interpret the keyword input. Locate the following code in the `setCatEntryParameters()` function:

```
//name
if (name != null && !name.equals("")) {
    catEntrySearchDB.setSearchTerm(name);
    if (nameOp.equals(SEARCH_OPERATOR_EQUAL)) {
        catEntrySearchDB.setSearchTermOperator(SEARCH_OPERATOR_EQUAL);
        catEntrySearchDB.setSearchTermCaseSensitive(SEARCH_CASE_SENSITIVE_YES);
        catEntrySearchDB.setSearchType("EXACT");
    } else {
        catEntrySearchDB.setSearchTermOperator(SEARCH_OPERATOR_LIKE);
        catEntrySearchDB.setSearchTermCaseSensitive(SEARCH_CASE_SENSITIVE_NO);
    }
}

// search scope
//By default in the catalog entry search databean, it is set to 1, which means
// search for name and short description
//Set it to a 2 means search on the name field only
// 1 = search on name + short description
// 2 = search on name only
if (new Boolean (searchScope).booleanValue())
{
    catEntrySearchDB.setSearchTermScope(new Integer (1));
} else {
    catEntrySearchDB.setSearchTermScope(new Integer (2));
}
}
```

Immediately under this section, add the following code:

```
// keyword. Type can be "ANY", "ALL", or "EXACT"
if (keyword != null && !keyword.equals("")) {
    catEntrySearchDB.setKeyword(keyword);
    if (keywordOp.equals(SEARCH_OPERATOR_EQUAL)) {
        catEntrySearchDB.setKeywordOperator(SEARCH_OPERATOR_EQUAL);
        catEntrySearchDB.setKeywordCaseSensitive(SEARCH_CASE_SENSITIVE_YES);
        catEntrySearchDB.setKeywordType("EXACT");
    } else {
        catEntrySearchDB.setKeywordOperator(SEARCH_OPERATOR_LIKE);
        catEntrySearchDB.setKeywordCaseSensitive(SEARCH_CASE_SENSITIVE_NO);
        catEntrySearchDB.setKeywordType("ANY");
    }
}
```

- j. Save the file, but do not close the development environment.
4. This procedure creates a new properties file, in a new folder, within the WebSphere Commerce development environment:
 - a. In the WebSphere Commerce development environment, navigate to the WebSphereCommerceServerExtensionsLogic project.
 - b. Navigate to the src folder.
 - c. Right-click on the src folder, and select **New**, and then **Package**.
 - d. In the **Name Field** on the New Java Package dialog, type a unique name. For the purposes of this scenario, type `com.myCompany.catalog.Properties`, where `myCompany` represents a custom directory name. Packages created by IBM, and included in WebSphere Commerce, follow a naming convention which begins with "com.ibm...". Click **Finish** to create the package.
 - e. Right-click on the `com.myCompany.catalog.properties` folder, and select **New**, and then **Other**. In the New dialog, click **Simple, File**, and then **Next**.
 - f. In the **File name** field, type a unique name. For the purposes of this scenario, type `CategoryNLS_locale.properties`, where `locale` represents the locale from which your business users will access the WebSphere Commerce Accelerator.
 - g. Click **Finish** to create the file and open it in an editor.
 - h. To simplify future maintenance, it is a good idea to include comments in the new file. These comments are optional, though strongly recommended. At the top of this file, include some comments similar to the following to clarify the file's purpose:

```
#
# Customized properties for catalogs
#####
categoryFindKeyword = Keyword
```

- i. Save the file, but do not close the development environment.

This step addresses the label in the language for the locale you specified why you created the file. If you wanted to enter a string in more than one language, you would have to perform this step for each language, creating a file for each corresponding locale.

If a business user attempts to access this page using a locale for which there is no explicit support, the tools framework will default to the `CategoryNLS.properties` file, which will not contain a string. You should ensure that you create a version of the properties file for every locale from which you expect users to access it.

5. When you create a custom properties file, you must update the appropriate `resources.xml` file so that the new file is available to the tools within the component. Updating the appropriate `resources.xml` file is not done within the development environment. To make this update, do the following:

- a. Copy the `WCDE_installdir/xml/tools/catalog/resources.xml` file, and move the file to the `myCustomXML/tools/catalog` directory.
- b. Open the new `resources.xml` file in a text editor.
- c. Scroll down to the `resourceBundle` section of the file, and update the code so that it matches the following sample:

```
<resourceBundle name="CategoryNLS">
  <bundle>com.ibm.commerce.tools.catalog.properties.CategoryNLS</bundle>
  <bundle>com.myCompany.catalog.Properties.CategoryNLS</bundle>
</resourceBundle>
```

where `myCompany` represents your custom directory name.

- d. Save the file.
6. In these four JSP files (`KitItemPickListLayout.jsp`, `KitLayout.jsp`, `MAssocLayout.jsp`, `ProductUpdateDetail.jsp`), update the paths to the `CatalogSearchUtil.jsp` file. Do the following:
 - a. Navigate to the `CommerceAccelerator` project.
 - b. Navigate to the `Web Content/tools/catalog` folder.
 - c. Right-click on the `KitItemPickListLayout.jsp` file, and select **Copy**.
 - d. Right-click on the `Web Content/tools/custom` folder, and select **Paste**.
 - e. Navigate to the `Web Content/tools/custom` folder.
 - f. Double-click the new file to open it in a text editor.
 - g. Search for `CatalogSearchUtil.jsp`. Replace the current `<%@include file=` path line with

```
<%@include file="../custom/CatalogSearchUtil.jsp" %>
```

- h. Save and close the file.
- i. Repeat these steps for the `KitLayout.jsp` file, `MAssocLayout.jsp` file, and `ProductUpdateDetail.jsp` file.

7. Register the `KitItemPickListLayout.jsp` file, `KitLayout.jsp` file, `MAssocLayout.jsp` file, and `ProductUpdateDetail.jsp` `CatalogSearchUtil.jsp` file into the `VIEWREG` table with the following SQL statements:

```
update viewreg set PROPERTIES='docname=tools/custom/KitItemPickListLayout.jsp'
where VIEWNAME = 'KitItemPickListView'
```

```
update viewreg set PROPERTIES='docname=tools/custom/KitLayout.jsp'
where VIEWNAME = 'KitLayoutView'
```

```
update viewreg set PROPERTIES='docname=tools/custom/MassocLayout.jsp'
where VIEWNAME = 'MassocLayoutView'
```

```
update viewreg set PROPERTIES='docname=tools/custom/ProductUpdateDetail.jsp'
where VIEWNAME = 'ProductUpdateDetail'
```

8. Creating a custom XML file in a custom directory requires that you update the instance `XMLPath` setting. This setting governs the locations in which the application will attempt to locate XML files. It functions in a manner similar to a Java classpath setting. This is not done within the development environment. To update the `XMLPath` setting, do the following:

- a. Change to the `WCDE_installdir/instances/instance_name/xml/` directory.
- b. Open the `instance_name.xml` file in a text editor.
- c. Scroll down to the

```
ToolsGeneralConfig
```

section of the file, and update the `XMLPath` by prepending the following to the value:

```
myCustomXML/tools;
```

where *myCustomXML* represents the absolute path to your custom directory, including the drive letter.

- d. Save the file.

Changing the XMLPath setting in the instance configuration file enable this customization only for the single instance. All other instances will not include this new button. If you have multiple instances to which you want to apply the customization, you must repeat this step for each instance.

Attention

Applying fix packs, or performing migration may overwrite any changes made to this file.

9. Test your customization in your development environment. To complete this test, do the following:
 - a. Stop and restart your development WebSphere Commerce instance. Refer to the *WebSphere Commerce Programming Guide and Tutorials* for details about how to stop and restart this instance.
 - b. Launch the WebSphere Commerce Accelerator.
 - c. From the **Business** **Product** or **Professional** **Merchandise** menu, select **Find Catalog Entries**. The new **Keyword** field should display on the Catalog Entry Search page.
 - d. Search for a catalog entry using the **Keyword** field. Choose a keyword not found in the catalog entry name or short description , such as an attribute (type of material, color, size, format). If you receive the correct results, then the customization has been a success, and you can proceed to propagate all of the changes you made to the development environment to the production environment as detailed in the following steps. If this search fails, you will have to determine the cause of the error, and debug.
10. Export the updated assets:
 - a. Navigate to the Web Content/tools/custom folder.
 - b. Right-click the CatEntrySearchDialog.jsp file and select **Export**. The Export wizard opens.
 - c. In the Export wizard, do the following:
 - 1) Select **File system** and click **Next**.
 - 2) Select the following JSP files:
 - CatEntrySearchDialog.jsp
 - CatalogSearchUtil.jsp
 - KitItemPickListLayout.jsp
 - KitLayout.jsp
 - MAssocLayout.jsp
 - ProductUpdateDetail.jsp
 - 3) Navigate to the WebSphereCommerceServerExtensionsLogic/src/com/*myCompany*/catalog/properties folder, and select the CategoryNLS_locale.properties file.
 - 4) Select **Create directory structure for selected files**.
 - 5) In the **To directory** field, type a temporary directory into which these resources will be placed. For example, type C:\ExportTemp\StoreAssets.

- 6) Click **Finish**.
 - 7) If prompted, create the specified directory.
11. Transfer the updated assets to the production environment. To transfer the files, do the following:
- a. Stop the WebSphere Commerce instance that is running within WebSphere Application Server. Refer to the *WebSphere Commerce Installation Guide* for your platform and database for details about how to stop this instance.
 - b. On the target machine, locate the WebSphere Commerce instance .ear directory. The following paths are an example of this directory:
 - ▶ 400 QIBM/ProdData/WebAsAdv4//installedApps/
WAS_node_name/ WC_instance_name.ear
 - ▶ AIX ▶ Linux ▶ Solaris WAS_installdir/installedApps/cellName/
WC_instance_name.ear
 - ▶ Windows WAS_installdir\installedApps\cellName\ WC_instance_name.ear
 where:
 - instance_name*
The name of your WebSphere Commerce instance.
 - ▶ 400 WAS_node_name
The iSeries™ system where the WebSphere Application Server product is installed.
 - cellName*
The WebSphere Application Server cell name.
 - c. Copy the files exported in step 10, into the following directories:
 - CatEntrySearchDialog.jsp**
WC_instance_name.ear/CommerceAccelerator.war/tools/custom
 - CatalogSearchUtil.jsp**
WC_instance_name.ear/CommerceAccelerator.war/tools/custom
 - KitItemPickListLayout.jsp**
WC_instance_name.ear/CommerceAccelerator.war/tools/custom
 - KitLayout.jsp**
WC_instance_name.ear/CommerceAccelerator.war/tools/custom
 - MAssocLayout.jsp**
WC_instance_name.ear/CommerceAccelerator.war/tools/custom
 - ProductUpdateDetail.jsp**
WC_instance_name.ear/CommerceAccelerator.war/tools/custom
 - CategoryNLS_locale.properties**
WC_instance_name.ear/properties/com/myCompany/catalog
/properties
 - d. Transfer the following file to your production environment:
 - WCDE_installdir/xml/tools/catalog/resources.xml
Copy to WC_installdir/xml/tools/catalog
 - e. Update the WC_installdir/instances/instance_name/xml/instance_name.xml file, so that it reflects the updated XMLPath value.
 - f. Propagate your changes to the VIEWREG table in the database to the production database.




- g. Restart your WebSphere Commerce instance. Refer to the *WebSphere Commerce Installation Guide* for your platform and database for details about how to start this instance.

Assets

The catalog search component is based upon the following assets.

JSP assets

The catalog JSP assets listed in the following table should be modified using the following procedure:

1. Open the WebSphere Commerce development environment (**Start > Programs > IBM WebSphere Commerce development environment**) and switch to the
 -  Business  Professional J2EE Perspective Project Navigator view
 -  Express Express Project Navigator view.
2. Navigate to the CommerceAccelerator project.
3. Navigate to the Web Content/tools/catalog directory.
4. Create a custom folder.
5. Copy the target file from the appropriate component folder to your custom folder.
6. Double-click the new file in the custom folder to open it in the editor window.
7. Update the file as necessary.
8. Update the VIEWREG database table to point to the customized version of the file.

The following table lists all of the catalog search JSP files.

Table 1. Catalog JSP assets

Page	JSP file	Properties file
Catalog Entry Search Page	CatEntrySearchDialog.jsp	<i>WC_installdir/instance.ear/properties/com/ibm/commerce/tools/catalog/properties/CategoryNLS_locale.properties</i>
Search Criteria collection	CatalogSearchUtil.jsp	

XML assets

The catalog search XML assets listed in the following table should be modified using the following procedure:

1. Create a new directory in which you will store your custom XML files.
2. Copy the XML file from your *WC_installdir/xml/tools/catalog* directory, into your custom directory.
3. Open the file in an editor and make any necessary changes.
4. Update the XMLPath value in the *instance_name.xml* file such that you prepend the custom directory path to the existing list of paths.

The following table lists all of the catalog search XML files that may require updates.

Table 2. Catalog XML files

XML file name	Description
resources.xml	Maps the different XML files used by the catalog to their name spaces.
catEntrySearchDialog	Defines the menu used on the Catalog Entry Search page.

Database assets

The catalog search component makes use of data stored in the database table described in the following table:

Table 3. Catalog database tables

Table name	Column name	Additional information
CATENTDESC	KEYWORD	The catalog entry keyword.

Chapter 3. Product Management tools tabs and columns

Architecture

The Product Line Update tool, also called the dynamic table, in WebSphere Commerce serves to create, update, delete and organize catalog entries. It consists of user interface JSP pages that provide a table for visually editing catalog entry data and a controller command that accepts XML input from the user interface and stores the data into the database.

The number of tabs, and columns inside a tab, is controlled by the CatalogPageList.xml file. The Product Management tools are constructed from the ProductUpdateDialog.jsp file, which is composed of three different frames:

- ProductUpdateTitle.jsp
- ProductUpdateDetail.jsp
- ProductUpdateBottom.jsp (which contains the URL command to call when saving the data)

The Product Management tools also use the CatalogCommonFunctions.js file, which provides the JavaScript™ functions needed by the tool.

Customization scenarios

Customizing tabs and columns

Important

This scenario and the previous keyword search scenario both use the same user interface field to complete the customization tasks. Each set of required changes cannot coexist. Only one customization scenario can be supported at a time.

You can customize the view tabs and columns for the Product Management tools pages. In this scenario, the Keyword column from the Images tab will be changed to contain medium sized images. Once the tasks are complete, the Images tab will have the following three columns, in this order:

- Thumbsize Image
- Medium Image
- Full Size Image

You will need to customize the CatalogPageList.xml file and the ProductNLS_locale.properties file.

1. Create a new directory. For the purposes of this scenario, name the directory */myCustomXML/*, where *myCustomXML* represents some a directory name.
2. Repeat this process until you have created the following path:
/myCustomXML/tools/catalog/
3. Update the Catalog Page List XML file to change the name and the order of the columns in the Images tab in the Product Management tools. To protect your customized data, it must be created in a safe place, separate from the WebSphere Commerce assets. This procedure creates a new Catalog Page List

definition file, in a new folder. This is not done within the development environment. To add the column, do the following:

- a. Change to the `WC_installdir/xml/tools/` directory.
- b. Create a new directory. For the purposes of this scenario, name the directory `myCatalog`.
- c. Make a copy of the `WC_installdir/xml/tools/catalog/CatalogPageList.xml` file. Move the file to the `/myCustomXML/tools/catalog/` directory.
- d. Open the `myCatalogPageList.xml` file in a text editor.
- e. Move the following line:

```
<column id="keyword" tabid="4" type="TEXTAREA" CLASS="dtable" input="STRING"
  maxsize="254" width="30%" title="productUpdateDetail_Keyword" other=""/>
```

between these two lines:

```
<column id="thumbnailImage" tabid="4" type="TEXTAREA" CLASS="dtable" input="STRING"
  maxsize="254" width="250px" title="productUpdateDetail_ThumbNail"
  other="ONCHANGE=fcnOnChangeThumbnail(this)"/>
```

```
<column id="fullImage" tabid="4" type="TEXTAREA" CLASS="dtable" input="STRING"
  maxsize="254" width="250px" title="productUpdateDetail_FullImage" other=""/>
```

- f. In the column `id="keyword"` segment, change
`title="productUpdateDetail_Keyword"`

to

```
title="productUpdateDetail_MediumImage"
```

You should see the following changes:

```
<column id="thumbnailImage" tabid="4" type="TEXTAREA" CLASS="dtable" input="STRING"
  maxsize="254" width="250px" title="productUpdateDetail_ThumbNail"
  other="ONCHANGE=fcnOnChangeThumbnail(this)"/>
```

```
<column id="keyword" tabid="4" type="TEXTAREA" CLASS="dtable" input="STRING"
  maxsize="254" width="30%" title="productUpdateDetail_MediumImage" other=""/>
```

```
<column id="fullImage" tabid="4" type="TEXTAREA" CLASS="dtable" input="STRING"
  maxsize="254" width="250px" title="productUpdateDetail_FullImage" other=""/>
```

- g. Save the file.

4. When you create a custom XML file to update the user interface, you must update the appropriate `resources.xml` file so that the new user interface displays as expected. This is not done within the development environment. To make this update, do the following:

- a. Change to the `WC_installdir/xml/tools/catalog` directory.
- b. Open the `resources.xml` file in an editor.
- c. Scroll down to the `resourceXML` section of the file, and update the catalog list page entry so that it matches the following sample:

```
<resourceXML name="CatalogPageList"
  file="myCustomXML/tools/catalog/CatalogPageList.xml"
```

- d. Save the file.

5. Override the existing `ProductNLS_en_US.properties` file to add the information for the changed columns in the Images tab. To protect your customized data from being overwritten during migration to a future version, or during installation of a future fix pack, or some similar event, it must be created in a

safe place, separate from the WebSphere Commerce assets. This procedure creates a new properties file, in a new folder, within the WebSphere Commerce development environment:

- a. Open the WebSphere Commerce development environment (**Start > Programs > IBM WebSphere Commerce development environment**) and switch to the

  J2EE Perspective Project Navigator view

 Express Project Navigator view.

- b. Navigate to the WebSphereCommerceServerExtensionsLogic project.
- c. Navigate to the src folder.
- d. Right-click on the src folder, and select **New**, and then **Package**.
- e. In the **Name Field** on the New Java Package dialog, type a unique name. For the purposes of this scenario, type `com.myCompany.catalog.Properties`, where *myCompany* represents a custom directory name. Packages created by IBM, and included in WebSphere Commerce, follow a naming convention which begins with "com.ibm...".
- f. Right-click on the `com.myCompany.catalog.properties` folder, and select **New**, and then **Other**. In the New dialog, click **Simple**, **File**, and then **Next**.
- g. In the **File name** field, type `ProductNLS_locale.properties`, where *locale* represents your locale from which your business users will access the WebSphere Commerce Accelerator.
- h. Click **Finish** to create the file and open it in a text editor.
- i. To simplify future maintenance, it is a good idea to include comments in the new file. This is optional, though strongly recommended. At the top of this file, include some comments similar to the following to clarify the purpose of the file:

```
#
# Customized properties for catalogs
# Changed columns for the Images tab: new order and replace keyword column with
# medium image
#####
```

- j. Add the following line:

```
productUpdateDetail_MediumImage=Medium Image
```

- k. Save the file.

6. When you create a custom properties file, you must update the appropriate `resources.xml` file so that the new file is available to the tools within the component. This is not done within the development environment. To make this update, do the following:


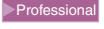






- a. Change to the `WC_installdir/xml/tools/catalog` directory.
- b. Open the `resources.xml` file in a text editor.
- c. Scroll down to the `resourceBundle` section of the file, and update the code so that it matches the following sample:

```
<resourceBundle name="ProductNLS"
  <bundle>com.ibm.commerce.tools.catalog.properties.ProductNLS</bundle>
  <bundle>com.myCompany.catalog.properties.myProductNLS</bundle>
</resourceBundle>
```

where *myCompany* represents your custom directory name.

- d. Save the file.

7. Test your customization in your development environment. To complete this test, do the following:

- a. Stop and restart your development WebSphere Commerce instance. Refer to the *WebSphere Commerce Programming Guide and Tutorials* for details about how to stop and restart this instance.
 - b. Launch the WebSphere Commerce Accelerator.
 - c. From the  **Product** or  **Merchandise** menu, select **Product Management**.
 - d. Click the Images tab. The three columns — Thumbsize Image, Medium Image, Full Size Image — should display in this order. If you receive the correct results, then the customization has been a success, and you can proceed to propagate all of the changes you made to the development environment to the production environment as detailed in the following steps. If this check fails, you will have to determine the cause of the error, and debug.
8. Export the updated assets:
 - a. Navigate to the `WebSphereCommerceServerExtensionsLogic/src/com/myCompany/catalog/properties` folder, and select the `ProductNLSlocale.properties` file.
 - b. Right-click the `ProductNLSlocale.properties` file and select **Export**. The Export Wizard opens. In the Export wizard, do the following:
 - 1) Select **Create directory structure for selected files**.
 - 2) In the **To directory** field, enter a temporary directory into which these resources will be placed. For example, enter `C:\ExportTemp\StoreAssets`.
 - 3) Click **Finish**.
 - 4) If prompted, create the specified directory.
 9. Transfer the updated assets to the production environment. To transfer the files, do the following:
 - a. Stop the WebSphere Commerce instance that is running within WebSphere Application Server. Refer to the *WebSphere Commerce Installation Guide* for your platform and database for details about how to stop this instance.
 - b. On the target machine, locate the WebSphere Commerce instance .ear directory. The following paths are an example of this directory:
 -  `QIBM/ProdData/WebAsAdv4/installedApps/WAS_node_name/WC_instance_name.ear`
 -    `/WAS_installdir/installedApps/cellName/WC_instance_name.ear`
 -  `WAS_installdir\installedApps\cellName\WC_instance_name.ear`
 where:
 - instance_name*
The name of your WebSphere Commerce instance.
 -  `WAS_node_name`
The iSeries system where the WebSphere Application Server product is installed.
 - cellName*
The WebSphere Application Server cell name.
 - c. Transfer the following files from your production environment:
 - `/myCustomXML/tools/catalog/CatalogPageList.xml`
Copy to `WC_installdir/xml/tools/custom`

WCDE_installdir/xml/tools/catalog/resources.xml

Copy to *WC_installdir/xml/tools/catalog*

- d. Update the *WC_installdir/instances/instance_name/xml/instance_name.xml*, so that it reflects the updated XMLPath value.
- e. Propagate your changes to the VIEWREG table in the database to the production database.
- f. Restart your WebSphere Commerce instance. Refer to the *WebSphere Commerce Installation Guide* for your platform and database for details about how to start this instance.

Supporting customized product information

This section adds support for catalog entry inventory information. The example illustrates how to customize the JSP files and create a new command to reuse the product line update command and fulfill additional tasks. Once completed, you will be able to update the catalog entry inventory for ItemBean and PackageBean.

1. Create a new directory.



If you completed the preceding scenario on customizing tabs and columns, then this directory will already exist.

For the purposes of this scenario, name the directory */myCustomXML/*, where *myCustomXML* represents some a directory name.

2. Repeat this process until you have created the following path:
/myCustomXML/tools/catalog/
3. Update the Catalog Page List XML file to add a new tab with two columns in the Product Management tools. To protect your customized data, it must be created in a safe place, separate from the WebSphere Commerce assets. This procedure creates a new Catalog Page List definition file, in a new folder. This is not done within the development environment. To add the column, do the following:
 - a. Change to the *WC_installdir/xml/tools/* directory.
 - b. Create a new directory. For the purposes of this scenario, name the directory *myCatalog*.
 - c. Make a copy of the *WC_installdir/xml/tools/catalog/CatalogPageList.xml* file and move it to the */myCustomXML/tools/catalog* directory.
 - d. Open the *CatalogPageList.xml* file in a text editor.
 - e. Locate the following section:

```
<tab id="Details"      title="ProductUpdateTitle_Tab_0"/>
<tab id="Category"    title="ProductUpdateTitle_Tab_1"/>
<tab id="Description" title="ProductUpdateTitle_Tab_2"/>
<tab id="Extended"    title="ProductUpdateTitle_Tab_3"/>
<tab id="Images"      title="ProductUpdateTitle_Tab_4"/>
<tab id="Custom"      title="ProductUpdateTitle_Tab_5"/>
```

Add the following line to create a new tab called **Inventory**:

```
<tab id="Inventory"    title="ProductUpdateTitle_Tab_6"/>
```

- f. Locate the following section:

```
<column id="customField5"      tabid="5" type="TEXTAREA" CLASS="dtable"
input="STRING" maxsize="254" width="40%" title="productUpdateDetail_Field5"
other="" />
```

Add the following lines to create two columns, called **Inventory Flag** and **Inventory Quantity**:

```
<column id="inventoryFlag"      tabid="6" type="TEXTAREA" CLASS="dtable"
      input="STRING" maxsize="64" width="50%" title="productUpdateDetail_InventoryFlag"
      other="ONCHANGE=fcnIntegerOnChange(this)" />

<column id="inventoryQuantity"  tabid="6" type="TEXTAREA" CLASS="dtable"
      input="STRING" maxsize="64" width="50%" title="productUpdateDetail_InventoryQuantity"
      other="ONCHANGE=fcnNumberOnChange(this)" />
```

- g. Save the file.
4. When you create a custom XML file to update the user interface, you must update the appropriate resources.xml file so that the new user interface displays as expected. This is not done within the development environment.



If you completed the preceding scenario on customizing tabs and columns, then this update will already exist.

To make this update, do the following:

- a. Change to the `WC_installdir/xml/tools/catalog` directory.
- b. Open the resources.xml file in an editor.
- c. Scroll down to the resourceXML section of the file, and update the catalog list page entry so that it matches the following sample:


```
<resourceXML name="CatalogPageList"
      file="/myCustomXML/tools/catalog/CatalogPageList.xml
```
- d. Save the file.
5. Modify the ProductNLS_en_US.properties file to add the information for the new tab and its columns:
 - a. Open the WebSphere Commerce development environment (**Start > Programs > IBM WebSphere Commerce development environment**) and switch to the

Business
Professional

J2EE Perspective Project Navigator view

Express Express Project Navigator view.
 - b. Navigate to the WebSphereCommerceServerExtensionsLogic project.
 - c. Navigate to the src folder.
 - d. Right-click on the src folder, and select **New**, and then **Package**.
 - e. In the **Name Field** on the New Java Package dialog, type a unique name. For the purposes of this scenario, type `com.myCompany.catalog.Properties`, where `myCompany` represents a custom directory name. Packages created by IBM, and included in WebSphere Commerce, follow a naming convention which begins with "com.ibm...".
 - f. Right-click on the `com.myCompany.catalog.properties` folder, and select **New**, and then **Other**. In the New dialog, click **Simple, File**, and then **Next**.
 - g. In the **File name** field, type `ProductNLS_locale.properties`, where `locale` represents your locale from which your business users will access the WebSphere Commerce Accelerator.



If you completed the preceding scenario on customizing tabs and columns, then this update will already exist.

- h. Click **Finish** to create the file and open it in a text editor.
- i. To simplify future maintenance, it is a good idea to include comments in the new file. This is optional, though strongly recommended. At the top of this file, include some comments similar to the following to clarify the purpose of the file:

```
#
# Customized properties for catalogs
# New inventory tab for the Product Management tools
#####
```

j. Add the following lines in the properties file:

```
ProductUpdateTitle_Tab_6=Inventory
productUpdateDetail_InventoryFlag=Inventory Flag
productUpdateDetail_InventoryQuantity=Quantity
```

k. Save the file, but do not close the development environment .

6. When you create a custom properties file, you must update the appropriate resources.xml file so that the new file is available to the tools within the component. This is not done within the development environment.



If you completed the preceding scenario on customizing tabs and columns, then this updated will already exist.

To make this update, do the following:

- a. Change to the *WC_installdir/xml/tools/catalog* directory.
- b. Open the resources.xml file in a text editor.
- c. Scroll down to the resourceBundle section of the file, and update the code so that it matches the following sample:

```
<resourceBundle name="ProductNLS">
  <bundle>com.ibm.commerce.tools.catalog.properties.ProductNLS</bundle>
  <bundle>com.myCompany.catalog.properties.ProductNLS</bundle>
</resourceBundle>
```

where *myCompany* represents your custom directory name.

- d. Save the file.
7. Update the createData(...) and createDataObject(...) methods in the Worksheet2.jsp file:

Important

After applying fix pack 2 (5.5.0.2), the file is renamed to Worksheet2.jspf. This scenario will refer to Worksheet2.jspf.

- a. Under the Web Content/tools directory, create a new folder called custom.
- b. In the WebSphere Commerce development environment, navigate to the CommerceAccelerator/Web Content/tools/catalog folder.
- c. Right-click on the Worksheet2.jspf file, and select **Copy**.
- d. Right-click on the Web Content/tools/custom folder and select **Paste**.
- e. Double-click the new file to open it in a text editor.
- f. In the createData(...) function, add the following section of code before the return results; line:

```
//Add this inventory section
try {

    InventoryAccessBean abInventory = new InventoryAccessBean();
    abInventory.setInitKey_catalogEntryId(abEntry.getCatalogEntryReferenceNumber());
    abInventory.setInitKey_fulfillmentCenterId(cmdContext.getStore().
        getFulfillmentCenterId());
    abInventory.setInitKey_storeId(cmdContext.getStoreId().toString());
    abInventory.refreshCopyHelper();

    result += "data["+index+"].inventoryFlag    = '" + toEscapeSlash(abInventory.
```

```

        getInventoryFlags()) + "';\n";
        result += "data["+index+"].inventoryQuantity = '" + toEscapeSlash(abInventory.
            getQuantity()) + "';\n";
    } catch (Exception e) {
    }
}

```

- g. In the function `createDataObject(...)` function, add these lines:

```

result += "    this.inventoryFlag    = '';\n";
result += "    this.inventoryQuantity = '';\n";

```

between the following lines:

```

result += "this.categoryID=E-1E;\n";

```

and

```

result += "};\n";

```

- h. Save the file.

8. In the `ProductUpdateDetail.jsp` file, make the following changes:

- In the WebSphere Commerce development environment, navigate to the `CommerceAccelerator/Web Content/tools/custom` folder.
- Right-click on the `ProductUpdateDetail.jsp` file, and select **Copy**.
- Right-click on the custom folder and select **Paste**.
- Double-click the new file to open it in a text editor.
- Add the import statement at the beginning of the `Worksheet2.jspf` file:

```

<%@ include file="../custom/Worksheet2.jspf" @>

```

- f. Save the file.

9. Create the `CatalogEntryXMLInventoryControllerCmdImpl` command implementation to process the customized data. To create this command implementation, do the following:

- Navigate to `WebSphereCommerceServerExtensionsLogic`.
- Right-click on the `src` folder, and select **New**, and then **Interface**.
- In the **Package** field, type `com.myCompany.commerce.tools.catalog.commands`.
- In the **Name** field, type `CatalogEntryXMLInventoryExtensionControllerCmd`.
- Click **Finish**. The file opens in a text editor.
- Add the following code to the file:

```

package com.myCompany.commerce.tools.catalog.commands;

import java.util.Hashtable;
import java.util.Vector;

import com.ibm.commerce.command.CommandFactory;
import com.ibm.commerce.datatype.TypedProperty;
import com.ibm.commerce.exception.ECApplicationException;
import com.ibm.commerce.exception.ECException;
import com.ibm.commerce.exception.ECSystemException;
import com.ibm.commerce. fulfillment.objects.InventoryAccessBean;
import com.ibm.commerce.ras.ECTrace;
import com.ibm.commerce.ras.ECTraceIdentifiers;
import com.ibm.commerce.tools.catalog.commands.CatalogEntryXMLControllerCmd;
import com.ibm.commerce.tools.catalog.commands.CatalogEntryXMLControllerCmdImpl;
import com.ibm.commerce.tools.xml.XMLStringReader;

/**
 * @author A developer
 *
 */
public class CatalogEntryXMLInventoryExtensionControllerCmdImpl {
    extends com.ibm.commerce.tools.command.ToolsControllerCommandImpl {

```

```

public static final String CLASSNAME =
    "com.myCompany.commerce.tools.catalog.commands.
        CatalogEntryXMLInventoryExtensionControllerCmdImpl";

private static final String CATALOGENTRY = "CatalogEntry";
private static final String CATALOGENTRY_ACTION = "action";
private static final String CATALOGENTRY_ACTIONUPDATE = "update";
private static final String CATALOGENTRY_CATALOGENTRYID = "catalogEntryId";
private static final String CATALOGINVENTORY = "Inventory";
private static final String CATALOGINVENTORY_FLAG = "inventoryFlag";
private static final String CATALOGINVENTORY_QUANTITY = "inventoryQuantity";

private Hashtable catalogEntryXML = null;

/*****
 * This method accepts input parameters from RequestProperties for the controller
 * command.
 *
 * @param reqParams com.ibm.commerce.datatype.TypedProperty
 * @exception ECAApplicationException.
 *
 *****/
public void setRequestProperties(TypedProperty reqParams)
    throws ECAApplicationException {
    super.setRequestProperties(reqParams);
    String xmlstring = requestProperties.getString("CatalogXML", null);
    if (xmlstring == null) {
        return;
    }

    try {
        XMLStringReader xsr = new XMLStringReader(xmlstring);
        Hashtable val = xsr.read();
        if (val != null) {
            catalogEntryXML = (Hashtable) val.get("XML");
        }
    } catch (ECSYSTEMException e) {
        ECTrace.trace(
            ECTraceIdentifiers.COMPONENT_TOOLSFRAMEWORK,
            this.getClass().getName(),
            "setRequestProperties",
            "Error parsing XML object");
    }
}

/*****
 * This function updates the catalog entry inventory information
 *
 * @param catalogEntryID String - catalog entry ID
 * @param h Hashtable - this is the hashtable of changes to the catalog entry
 *         inventory information
 * @throws ECAException
 *****/
public void updateCatalogEntryInventory(String catalogEntryID, Hashtable h)
    throws ECAException {
    final String METHODNAME = "updateCatalogInventory";

    try {
        //Update the access bean
        InventoryAccessBean abInventory = new InventoryAccessBean();
        abInventory.setInitKey_catalogEntryId(catalogEntryID);
        abInventory.setInitKey_fulfillmentCenterId(getCommandContext().getStore().
            getFulfillmentCenterId());
        abInventory.setInitKey_storeId(getCommandContext().getStoreId().toString());
        abInventory.refreshCopyHelper();
        if (h.containsKey(CATALOGINVENTORY_FLAG))

```

```

    {
        abInventory.setInventoryFlags(h.get(CATALOGINVENTORY_FLAG).toString());
    }
    if (h.containsKey(CATALOGINVENTORY_QUANTITY))
    {
        abInventory.setQuantity(h.get(CATALOGINVENTORY_QUANTITY).toString());
    }
    abInventory.commitCopyHelper();
} catch (Exception ex) {
    ECTrace.trace(
        ECTraceIdentifiers.COMPONENT_CATALOG,
        CLASSNAME,
        METHODNAME,
        "Cannot update Inventory.");
}
}

/*****
* This function updates a single catalog entry
*
* @param hEntry Hashtable - a hashtable with containing the catalog entry
* information
*****/
public void updateCatalogEntryObject(Hashtable hEntry) {
    final String METHODNAME = "updateCatalogEntryObject";

    try {
        String action = (String) hEntry.get(CATALOGENTRY_ACTION);
        String catalogEntryID =
            (String) hEntry.get(CATALOGENTRY_CATALOGENTRYID);
        if (action.equals(CATALOGENTRY_ACTIONUPDATE))
        {
            updateCatalogEntryInventory(catalogEntryID,hEntry);
        }
    } catch (Exception e) {
        ECTrace.trace(
            ECTraceIdentifiers.COMPONENT_CATALOG,
            CLASSNAME,
            METHODNAME,
            "Catalog Entry Inventory Update Failed.");
    }
}

/*****
* This function accepts an XML string parameter containing the update
* information for the set of catalog entries modified by the Product Line
* Update tool.
* These updates are then processed.
*
* @throws ECEException
*****/
public void performExecute()
throws com.ibm.commerce.exception.ECEException {
    final String METHODNAME = "performExecute";
    // perform server side parameter checking
    super.performExecute();

    // determine if there is something to parse
    if (catalogEntryXML != null) {
        java.lang.Object objXML = catalogEntryXML.get(CATALOGENTRY);
        if (objXML instanceof Hashtable) {
            updateCatalogEntryObject((Hashtable) objXML);
        }
        if (objXML instanceof Vector) {
            for (int i = 0; i < ((Vector) objXML).size(); i++) {
                Hashtable hXML = (Hashtable) ((Vector) objXML).elementAt(i);
            }
        }
    }
}

```

```

        updateCatalogEntryObject(hXML);
    }
}

try {
    //
    // Call the old command.
    // We do not need to know what it accepts but rather
    // just call it with the input from the JSP file like we did originally.
    //
    // Retrieve the response properties from the command since we don't know
    // what it sends back to the JSP file.
    //
    CatalogEntryXMLControllerCmd catalogEntryXMLControllerCmd = null;
    catalogEntryXMLControllerCmd =
        (CatalogEntryXMLControllerCmd) CommandFactory.createCommand(
            "com.ibm.commerce.tools.catalog.commands.CatalogEntryXMLControllerCmd",
            getStoreId());
    catalogEntryXMLControllerCmd.setCommandContext(getCommandContext());
    catalogEntryXMLControllerCmd.setRequestProperties(
        requestProperties);
    catalogEntryXMLControllerCmd.performExecute();
    responseProperties =
        catalogEntryXMLControllerCmd.getResponseProperties();
} catch (Exception e) {
    ECTrace.trace(
        ECTraceIdentifiers.COMPONENT_CATALOG,
        CLASSNAME,
        METHODNAME,
        "The update failed.");
}
}
}
}

```

- g. Save the file, but do not close the development environment.
10. Create the `CatalogEntryXMLInventoryExtensionControllerCmd` command. In the WebSphere Commerce development environment, do the following:
- Navigate to the `WebSphereCommerceServerExtensionsLogic` project.
 - Navigate to the `src` folder.
 - Right-click on the `com.myCompany.commerce.tools.catalog.commands` package, and select **New**, and then **Class**.
 - In the **Name** field, type `CatalogEntryXMLInventoryExtensionControllerCmd`.
 - Click **Finish** to open the command in a text editor.
 - Add the following code:

```

package
com.myCompany.commerce.tools.catalog.commands;
import
com.ibm.commerce.command.ControllerCommand;
/**
 * @author A developer
 *
 */

public interface CatalogEntryXMLInventoryExtensionControllerCmd extends
    ControllerCommand {

/**
 * Method interface name

```

```

*/

public static final String NAME = "com.myCompany.commerce.tools.catalog.
  commands.CatalogEntryXMLInventoryExtensionControllerCmd";

/**
 * Method implementation classname
 */

public static String defaultCommandClassName = "com.myCompany.commerce.tools.
  catalog.commands.CatalogEntryXMLInventoryExtensionControllerCmdImpl";
}

```

g. Save and close the file.

11. Create a view in the VIEWREG database table for the new JSP file. Run the following SQL statement:

```

update VIEWREG set PROPERTIES='docname=tools/custom/ProductUpdateDetail.jsp' where
  VIEWNAME='ProductUpdateDetail'



```

12. Register the command in the URLREG database table using the following SQL statement:

```

update urlreg set interfacename='com.myCompany.commerce.tools.catalog.
  commands.CatalogEntryXMLInventoryExtensionControllerCmd' where
  URL='CatalogEntryXMLControllerCmd'

```

13. Test your customization in your development environment. To complete this test, do the following:
 - a. Stop and restart your development WebSphere Commerce instance. Refer to the *WebSphere Commerce Programming Guide and Tutorials* for details about how to stop and restart this instance.
 - b. Launch the WebSphere Commerce Accelerator.
 - c. From the  **Business** **Product** or  **Professional** **Merchandise** menu, select **Product Management**.
 - d. Click the Inventory tab. If you receive the correct results, then the customization has been a success, and you can proceed to propagate all of the changes you made to the development environment to the production environment as detailed in the following steps. If this check fails, you will have to determine the cause of the error, and debug.
14. Export the updated assets:
 - a. Right-click the ProductUpdateDetail.jsp file and select **Export**. The Export wizard opens.
 - b. In the Export wizard, do the following:
 - 1) Select **File system** and click **Next**.
 - 2) Select the ProductUpdateDetail.jsp file.
 - 3) Select the Worksheet2.jpsf file.
 - 4) Navigate to the WebSphereCommerceServerExtensionsLogic/src/com/myCompany/catalog/properties folder, and select the ProductNLS_en_US.properties file.
 - 5) Navigate to the WebSphereCommerceServerExtensionsLogic/src/com/myCompany/commerce/tools/catalog/commands folder, and select the CatalogEntryXMLInventoryExtensionControllerCmd.java and CatalogEntryXMLInventoryExtensionControllerCmdImpl.java files.
 - 6) Select **Create directory structure for files**.




- 7) In the **To directory** field, type a temporary directory into which these resources will be placed. For example, type `C:\ExportTemp\StoreAssets`.
 - 8) Click **Finish**.
 - 9) If prompted, create the specified directory.
15. Transfer the updated assets to the production environment. To transfer the files, do the following:
- a. Stop the WebSphere Commerce instance that is running within WebSphere Application Server. Refer to the *WebSphere Commerce Installation Guide* for your platform and database for details about how to stop this instance.
 - b. On the target machine, locate the WebSphere Commerce instance .ear directory. The following paths are an example of this directory:
 - ▶ 400 QIBM/ProdData/WebAsAdv4/installedApps/ WAS_node_name/ WC_instance_name.ear
 - ▶ AIX ▶ Linux ▶ Solaris WAS_installdir/installedApps/cellName/ WC_instance_name.ear
 - ▶ Windows WAS_installdir\installedApps\cellName\ WC_instance_name.ear
 where:
 - instance_name*
The name of your WebSphere Commerce instance.
 - ▶ 400 WAS_node_name
The iSeries system where the WebSphere Application Server product is installed.
 - cellName*
The WebSphere Application Server cell name.
 - c. Copy the files exported in step 14, into the following directories:
 - ProductUpdateDetail.jsp**
WC_instance_name.ear/CommerceAccelerator.war/tools/custom
 - Worksheet2.jspf**
WC_instance_name.ear/CommerceAccelerator.war/tools/custom
 - ProductNLS_locale.properties**
WC_instance_name.ear/properties/com/myCompany/catalog/properties
 - CatalogEntryXMLInventoryExtensionControllerCmd**
com.myCompany.commerce.tools.catalog.commands
 - d. Transfer the following files from your production environment:
 - /myCustomXML/tools/catalog/CatalogPageList.xml*
Copy to WC_installdir/xml/tools/custom
 - WCDE_installdir/xml/tools/catalog/resources.xml*
Copy to WC_installdir/xml/tools/catalog
 - e. Update the WC_installdir/instances/instance_name/xml/instance_name.xml, so that it reflects the updated XMLPath value.
 - f. Propagate your changes to the VIEWREG table in the database to the production database.
 - g. Restart your WebSphere Commerce instance. Refer to the *WebSphere Commerce Installation Guide* for your platform and database for details about how to start this instance.

Assets

The Product Management tools component is based upon the following assets.

JSP assets

The Product Management tools JSP assets listed in the following table should be modified using the following procedure:

1. Open the WebSphere Commerce development environment (**Start > Programs > IBM WebSphere Commerce development environment**) and switch to the   J2EE Perspective Project Navigator view  Express Project Navigator view.
2. Navigate to the CommerceAccelerator project.
3. Navigate to the Web Content/tools/catalog directory.
4. Create a custom folder.
5. Copy the target file from the appropriate component folder to your custom folder.
6. Double-click the new file in the custom folder to open it in the editor window.
7. Update the file as necessary.
8. Update the VIEWREG database table to point to the customized version of the file.

The following table lists all of the catalog JSP files.

Table 4. Product Management tools JSP assets

Page	JSP	Properties file
Product Update tool	ProductUpdateDialog.jsp	WAS_installdir/installedApps/ properties/com/ibm/commerce/ tools/catalog/properties/ ProductNLS_locale.properties.
Title frame (menu)	ProductUpdateTitle.jsp	
Detail frame (table with the catalog entries)	ProductUpdateDetail.jsp	
Bottom frame (save and return buttons)	ProductUpdateBottom.jsp	
SKU Update tool	ItemUpdateDialog.jsp	
Detail frame (table with the catalog entries)	ItemUpdateDetail.jsp	
Bottom frame (save and return buttons)	ItemUpdateBottom.jsp	
iframe used by Edit > Find	ProductUpdateFind.jsp	
iframe used by Edit > Replace	ProductUpdateReplace.jsp	
iframe used when double-clicking a text area	ProductUpdateTextarea.jsp	
iframe to create the dialog to change a category	ProductCategoryDialog.html	
Dialog category tree frame to change a category	ProductCategoryDetail.jsp	
Dialog buttons frame to change a category	ProductCategoryButtons.jsp	

The JSP assets use the CatalogCommonFunction.js Javascript file.

The following Java assets belong to the com.ibm.commerce.tools.catalog.commands package:

- CatalogEntryXMLControllerCmd.java
- CatalogEntryXMLControllerCmdImpl.java

XML assets

The Product Management tools XML assets listed in the following table should be modified using the following procedure:

1. Create a new directory in which you will store your custom XML files.
2. Copy the XML file from your *WC_installdir/xml/tools/catalog* directory, into your custom directory.
3. Open the file in an editor and make any necessary changes.
4. Update the XMLPath value in the *instance_name.xml* file such that you prepend the custom directory path to the existing list of paths.

The following table lists all of the Product Management tools XML files that may require updates.

Table 5. Product Management tools XML files

XML file name	Description
resources.xml	Maps the different XML files used by the catalog to their name spaces.
CatalogMenuProduct.xml	Defines the menu used on the Product Update tool for the following catalog entries: products, bundles, and kits.
CatalogMenuItem.xml	Defines the menu used on the Product Update tool for SKUs.
CatalogPageList.xml	Defines the tabs and columns in the Product Update Tool.

Database assets

The Product Management tools component makes use of data stored in the database table described in the following table:

Table 6. Product Management tools database tables

Table name	Additional information
CATENTDESC	Catalog entry descriptions.
CATENTRY	All information related to a catalog entry.

Notices

Note to U.S. Government Users — — Documentation relating to restricted rights — — Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM product, program, or service may be used. Any functionality equivalent to product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering the subject matter in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chrome, Minato-ku
Tokyo 1061, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web

sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independent created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Lab Director
IBM Canada Ltd. Laboratory
8200 Warden Avenue
Markham, Ontario
L6G 1C7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This document may contain information about other companies' products, including references to such companies' Internet sites. IBM has no responsibility for the accuracy, completeness, or use of such information.

This product is based on the SETTM protocol.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to

IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 1996, 2003. All rights reserved.

Trademarks

The IBM logo and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries or both:

- IBM
- WebSphere

Solaris, Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft[®], Windows[®], and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction[™] LLC.

UNIX[®] is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

IBM