IBM WebSphere Commerce

Customizing Campaigns in WebSphere Commerce Accelerator

Version 5.5

IBM WebSphere Commerce

Customizing Campaigns in WebSphere Commerce Accelerator

Version 5.5

Note:
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 27.

First Edition, March 2004

This edition applies to IBM WebSphere Commerce Business Edition Version 5.5, IBM WebSphere Commerce Professional Edition Version 5.5, and IBM WebSphere Commerce - Express Version 5.5 (product number 5724-A18), 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.

Contents

About this book

Updates to this book

The latest version of this book, is available as a PDF file from the IBM® WebSphere® Commerce technical library Web site:

`www.ibm.com/software/commerce/library/`

Conventions used in this book

This book uses the following highlighting conventions:

Boldface type	Indicates commands or graphical user interface (GUI) controls such as names of fields, icons, or menu choices.
`Monospace type`	Indicates examples of text you enter exactly as shown, file names, and directory paths and names.
Italic type	Used to emphasize words. Italics also indicate names for which you must substitute the appropriate values for your system.

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 guide 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:

AIX `/usr/WebSphere/CommerceServer55`

400 `/QIBM/ProdData/CommerceServer55`

Linux `/opt/WebSphere/CommerceServer55`

Solaris `/opt/WebSphere/CommerceServer55`

Windows 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:

AIX /usr/WebSphere/AppServer

400 /QIBM/ProdData/WebAS5

Linux /opt/WebSphere/AppServer

Solaris /opt/WebSphere/AppServer

Windows C:\Program Files\WebSphere\AppServer

WCDE_installdir
This is the installation directory for either the WebSphere Commerce Studio, or the WebSphere Commerce - Express development environment.

The default installation directory for WebSphere Commerce Studio is `C:\WebSphere\CommerceStudio55`.

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

Knowledge requirements

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

- Blaze Advisor rule technology
- HTML, JavaScript™, and XML
- Structured Query Language (SQL)
- Java™ Programming
- JavaServer Pages technology
- WebSphere Commerce Studio or WebSphere Commerce - Express Developer Edition

Please refer to the *WebSphere Commerce Programming Guide and Tutorials*for more information on customizing WebSphere Commerce. This book is available from the following Web site:

`www.ibm.com/software/commerce/library`

Part 1. Customizing the Campaign components in the WebSphere Commerce Accelerator

This document discusses how to customize the Campaigns components in the WebSphere Commerce Accelerator. 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:

- Campaigns
- Ad copy

All of these components are tightly integrated to deliver marketing campaigns using WebSphere Commerce. The user interfaces for these components, as found in the WebSphere Commerce Accelerator facilitate your business user's ability to create and maintain the pieces that comprise a campaign within WebSphere Commerce.

Chapter 1. Campaigns

Architecture

The campaigns component in WebSphere Commerce, when considered without the associated components, serves to organize campaign initiatives, and act as a method by which to sort statistical information associated with marketing efforts.

Customization scenarios

Since the campaigns component has little effect on the functionality of your marketing efforts, it is not a component for which you will likely find a lot of need for customization. The scenario detailed below is a generic example of how you can add an action button to a list page. While the details of this customization scenario are specific to the campaigns components, you can easily adapt it to other components, with different targets by making the changes to the appropriate assets.

Adding an action button to the Campaigns list page

This scenario adds an action button to the Campaigns list page, which links directly to the campaign initiative wizard. This is useful insofar as it provides another way of accessing this tool.

1. Create a label for the new button. WebSphere Commerce design separates program logic from displayed strings, so labels for buttons, and all other interface elements are stored in properties files, which are also referred to as resource bundles. 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

      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, enter a unique name. For the purposes of this scenario, enter `com.myCompany.campaigns.properties`, where *myCompany* represents some custom directory name. Packages created by IBM, and included in WebSphere Commerce follow a naming convention which begins, "com.ibm..."
   f. Click **Finish** to create the package.
   g. Right-click on the com.*myCompany*.campaigns.properties folder, and select **New**, and then **Other**. In the New dialog, click **Simple**, **File**, and then **Next**.

h. In the **File name** field, enter a unique name. For the purposes of this scenario, enter CampaignsRB_*locale*.properties, where *locale* represents the locale from which your business users will access the WebSphere Commerce Accelerator.

i. Click **Finish** to create the file and open it in an editor.

j. To simplify future maintenance, it is a good idea to include comments in this new file. This is optional, though strongly recommended. At the top of this file, include some comments similar to the following to clarify this file's purpose:

```
#
# Customized properties for Campaigns
#
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
```

k. Scroll down and create a "Button definitions" section for the file, and insert a label for your new button. This should look similar to the following sample:

```
#
# Button definitions
#

newCustomInitiative = Create Initiative
```

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

This addresses the label in the language for the locale you specified when 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 CampaignsRB.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.

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

b. Repeat this process until you have created the following path:

   /*myCustomXML*/tools/campaigns/

   where *myCompany* represents some custom directory name.

c. Copy the /*WCDE_installdir*/xml/tools/campaigns/resources.xml file, and move the file to the /*myCustomXML*/tools/campaigns/ directory.

d. Open the new resources.xml file in an editor.

e. Scroll down to the "Resource bundle" section of the file, and update the code so that it matches the following sample:

```
<resourceBundle name="campaignsRB">
    <bundle>com.ibm.commerce.tools.campaigns.properties.CampaignsRB</bundle>
    <bundle>com.myCompany.campaigns.properties.CampaignsRB</bundle>
</resourceBundle>
```

   where *myCompany* represents your custom directory name.

f. Save the file.

3. Update the Campaign List definition XML file to add the new button, **Create Initiative**. Again, to protect your customized data, it must be created in a safe place, separate from the WebSphere Commerce assets. This procedure creates a new Campaign List XML definition file, in the same */myCustomXML*/tools/campaigns/ folder used in step 2. This is not done within the development environment. To add the button, do the following:
   a. Change to the */WCDE_installdir*/xml/tools/campaigns directory.
   b. Copy */WCDE_installdir*/xml/tools/campaigns/CampaignList.xml to the new */myCustomXML*/tools/Campaigns/ directory. Do not change the file name.
   c. Open the CampaignList.xml file in an editor.
   d. Scroll down to the code segment about buttons. Add the following code:

```
<menu name="newInitiative"
   action="basefrm.newInitiative()"
   selection="single" />
```

      Note that the value for the name attribute must match the value of the key that we just inserted into the properties file in step 1. Also, the action matches the action defined in the Initiative list page's corresponding XML file, also located in the */WCDE_installdir*/xml/tools/campaigns directory. This button performs the same function as the Create button on that page. Finally, we have set the value for the selection attribute to 'single', which means that the action is only available when a single campaign is selected. In other situations this value could be set to either 'none', or 'multiple', which would specify that the button was only available when either no campaigns were selected, or when more than one campaign was selected, respectively.
   e. Save the file.
4. When you create a custom XML file to update the 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 */myCustomXML*/tools/campaigns/ directory.
   b. Open the resources.xml file in an editor.
   c. Scroll down to the "XML file mappings" section of the file, and update the campaign list entry so that it matches the following sample:

```
<resourceXML name="CampaignList"
        file="/myCustomXML/tools/campaigns/CampaignList.xml" />
```

      where *myCustomXML* represents your custom directory name.
   d. Save the file.
5. 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 an 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.

6. Update the CampaignList.jsp with a new algorithm to handle the new button. To protect the custom code, this procedure creates a new JSP which contains the logic:
   a. In the WebSphere Commerce development environment, navigate to the CommerceAccelerator/Web Content/tools directory.
   b. Right-click on the folder, and select **New,** and then **Folder**. In the **Folder name** field, enter `custom`.
   c. Navigate to the Web Content/tools/campaigns folder.
   d. Right-click on the CampaignList.jsp file, and select **Copy**.
   e. Right-click on the Web Content/tools/custom folder, and select **Paste**.
   f. Navigate to the Web Content/tools/custom folder.
   g. Double-click the new file to open it in an editor.
   h. Update your JSP. Scroll down to the section of the file in which the JavaScript functions are defined, and update it so that it contains the following JavaScript (lines may be split here for presentation purposes):

```
function newInitiative () {
 var url = "<%= CampaignConstants.URL_CAMPAIGN_INITIATIVE_WIZARD_VIEW %>";
 if (top.setContent) {
  top.setContent("<%= UIUtil.toJavaScript((String)campaignsRB.get
         (CampaignConstants.MSG_CREATE_INITIATIVE)) %>", url, true);
 }
 else {
  parent.location.replace(url);
 }
}
```

This code can be found in the InitiativeList.jsp file in the same directory, and copied from that source.

   i. Also, because the custom file is in a different location, you have to update the following path in the JSP to reflect the original location. Search for the following include statement:

```
<%@ include file="common.jsp" %>
```

   and change it to:

```
<%@ include file="../campaigns/common.jsp" %>
```

   j. Save the file, but do not close the development environment.
7. Update the view that corresponds to the page in the VIEWREG database table. This table governs the relationships between stores, views, and the JSP files they launch. Because this customization created a new page for an existing

view, you must make a corresponding change in this table. To update the VIEWREG table, use the following SQL statement:

```
update VIEWREG set PROPERTIES = 'docname=tools/custom/CampaignList.jsp'
  where (VIEWNAME, STOREENT_ID) = (CampaignListView, 0)
```

8. 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 **Marketing** menu, select **Campaigns**. The new button should display on this page.
   d. Select a campaign, and click the new **Create Initiative** button. This should launch the Campaign initiative wizard. If this works, 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 fails, you will have to determine the cause of the error, and debug.
9. Export the updated assets:
   a. Right-click the new CampaignList.jsp file and select **Export**. The Export Wizard opens.

      In the Export wizard, do the following:
      1) Select **File system** and click **Next**.
      2) The CampaignList.jsp is selected.
      3) Navigate to the WebSphereCommerceServerExtensionsLogic/src/com/*myCompany*/campaigns/properties folder, and select the myCampaignsRB_*locale*.properties file.
      4) Select **Create directory structure for selected files**.
      5) In the **To directory** field, enter a temporary directory into which these resources will be placed. For example, enter `C:\ExportTemp\StoreAssets`
      6) Click **Finish**.
      7) If prompted, create the specified directory.
10. 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 is 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 9, above, into the following directories:

CampaignList.jsp
WC_*instance_name*.ear/CommerceAccelerator.war/tools/custom

myCampaignsRB_*locale*.properties
WC_*instance_name*.ear/properties/com/*myCompany*/campaigns/properties

d. Transfer the following files to your production environment:

/*myCustomXML*/tools/campaigns/CampaignList.xml
Copy to WC_*installdir*/xml/tools/custom

/*WCDE_installdir*/xml/tools/campaigns/resources.xml
Copy to WC_*installdir*/xml/tools/campaigns

e. Update the /*WC_installdir*/instances/*instance_name*/xml/*instance_name*.xml, so that it reflects the updated XMLPath value.

f. Propagate your change 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 campaign component is based upon the following assets.

JSP assets

The campaign 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/ 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 campaign JSP files.

Table 1. Campaign JSP Assets

Page	JSP	Properties file
List	CampaignList.jsp	*/WCDE_installdir*/properties/runtime/com/ibm /commerce/tools/campaigns/properties /CampaignsRB_*locale*.properties
General	CampaignGeneralPanel.jsp	
Business Objectives	CampaignBusinessPanel.jsp	
Summary	CampaignSummaryPanel.jsp	

XML assets

The campaign 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 */WCDE_installdir*/xml/tools/campaigns 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 campaign XML files that may require updates.

Table 2. Campaign XML files

XML filename	Description
CampaignList.xml	Defines the framework for the Campaign List page. This includes navigation, and action buttons, but the content in the primary frame originates in the CampaignList.jsp file.
CampaignNotebook.xml	Defines the framework for the Campaign Notebook. This includes navigation, and action buttons, but the content in the primary frame originates in the CampaignGeneralPanel.jsp and CampaignBusinessPanel.jsp files.
CampaignsDeletedDialog.xml	Defines the framework and content for the Campaign Deleted dialog.
CampaignSummaryDialog.xml	Defines the framework and content for the Campaign Summary dialog.
CampaignWizard.xml	Defines the framework for the Campaign Wizard. This includes navigation, and action buttons, but the content in the primary frame originates in the CampaignGeneralPanel.jsp and CampaignBusinessPanel.jsp files.

Database assets

The campaign component makes use of data stored in the database table described in the following table.

Table 3. Campaign database tables

Table name	Column name	Additional information
Campaign	CAMPAIGN_ID	System generated.
	DESCRIPTION	Entered by a business user on the General page.
	FIELD1	Available for custom information.
	LASTUPDATE	System generated.
	LASTUPDATEDBY	System generated.
	NAME	Entered by a business user on the General page.
	OBJECTIVE	Entered by a business user on the Business Objectives page.
	OWNER	Entered by a business user on the Business Objectives page.
	STATUS	Updated according to action buttons on the Campaigns list page.
	STOREENT_ID	Reference to STOREENT table.
	TYPE	Available for custom information.

Chapter 2. Ad copy

Architecture

The ad copy component in WebSphere Commerce acts as a repository for any marketing collateral, including images and flash animations, that is designed to support a campaign. Ad copy displays in the space defined by an e-Marketing Spot, and is obtained from the commerce database when the commerce server renders the page.

Each ad copy is associated with a click action, which is a URL that specifies an action to perform when a customer clicks on the ad copy.

Customization scenarios

Consolidating the wizard and notebook into single pages

This scenario streamlines the process of updating ad copy, by merging the general definition and description definition pages into a single page. This eliminates the need to navigate between the pages when creating and updating ad copy definitions. The existing ad copy pages are used by both the Ad Copy wizard, and the Ad Copy notebook depending on the action button clicked by the user. Therefore, we will create both, a new wizard, and a new notebook that reference the consolidated page.

To consolidate the ad copy tools, do the following:

1. Start this customization by creating the consolidated page. To create this page, do the following:
   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 CommerceAccelerator project.
   c. Navigate to the Web Content/tools folder.
   d. Right-click on folder, and select **New,** and then **Folder**. In the **Folder name** field, enter `custom`.

      Note: If you completed "Adding an action button to the Campaigns list page" on page 3, this folder will already exist.
   e. Navigate to the Web Content/tools/campaigns directory.
   f. Double-click the CollateralGeneralPanel.jsp and CollateralDescriptionPanel.jsp files to open them in the editor window. Since you are merging the two panels, the idea here is to update the general panel JSP file with the code that constitutes the description JSP, and then save the new JSP to the new custom folder. Since this scenario combines the two pages, save the CollateralGeneralPanel.jsp file in the new custom folder, using a new name, for example, `ConsolidatedCollateralPanel.jsp`.
   g. Scroll down to the HTML `<title>` element, and replace it with the following code:

```
<TITLE><%= campaignsRB.get("ConsolidatedCollateralNavigation") %></TITLE>
```

This hard codes a string variable name as the title, which is used to generate the notebook table of contents. Later in the scenario, the string variable will be defined in a custom properties file.

h. Scroll down to the HTML <H1> element, and replace it with the following code:

```
<H1><%= campaignsRB.get("ConsolidatedCollateralPanel") %></H1>
```

This hard codes a string variable name as the page heading. Later in the scenario, the string variable will be defined in a custom properties file.

i. Update the `parent.get` segment of the LoadPanelData function so that it looks like the following:

Note: This code sample below was created by copying code from the CollateralDescriptionPanel.jsp file, and merging it with code originally located in the CollateralGeneralPanel.jsp. No new code was required.

```
if (parent.get) {
   var o = parent.get("<%= CampaignConstants.ELEMENT_COLLATERAL %>", null);
   if (o != null) {
      loadValue(<%= CampaignConstants.ELEMENT_COLLATERAL_NAME %>,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_NAME %>);
      loadValue(<%= CampaignConstants.ELEMENT_COLLATERAL_URL %>,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL %>);
      loadValue(<%= CampaignConstants.ELEMENT_COLLATERAL_FIELD1 %>,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_FIELD1 %>);
      loadValue(<%= CampaignConstants.ELEMENT_COLLATERAL_FIELD2 %>,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_FIELD2 %>);
      setOptionSelected(<%= CampaignConstants.ELEMENT_COLLATERAL_TYPE %>,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_TYPE %>);
      urlSelection = o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND_TYPE %>;
      loadValue(<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND %>,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND %>);
      loadValue(<%= CampaignConstants.ELEMENT_COLLATERAL_URL_CUSTOM %>,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_CUSTOM %>);
      loadValue(<%= CampaignConstants.ELEMENT_COLLATERAL_LOCATION %>,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_LOCATION %>);
      loadValue(<%= CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %>,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %>);
      loadValue(<%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %>,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %>);
      loadValue(<%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %>,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %>);

      if (o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_TYPE %> == "custom") {
         loadRadioValue(<%= CampaignConstants.ELEMENT_COLLATERAL_URL_TYPE %>, "false");
      }
      else {
         loadRadioValue(<%= CampaignConstants.ELEMENT_COLLATERAL_URL_TYPE %>, "true");
      }

      if ((o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND_TYPE %> ==
              "<%= CampaignConstants.URL_PRODUCT_DISPLAY %>") ||
         (o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND_TYPE %> ==
              "<%= CampaignConstants.URL_ORDER_ITEM_ADD %>") ||
         (o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND_TYPE %> ==
              "<%= CampaignConstants.URL_INTEREST_ITEM_ADD %>")) {
            loadValue(urlCommandProduct,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND_PARAMETER %>);
      }
      else if (o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND_TYPE %> ==
              "<%= CampaignConstants.URL_CATEGORY_DISPLAY %>") {
         loadValue(urlCommandCategory,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND_PARAMETER %>);
      }
      else if (o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND_TYPE %> ==
              "<%= CampaignConstants.URL_ACCEPT_COUPON %>") {
         loadValue(urlCommandCoupon,
              o.<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND_PARAMETER %>);
      }
   }

   if (parent.get("<%= CampaignConstants.ELEMENT_COLLATERAL_NAME_REQUIRED %>", false)) {
      parent.remove("<%= CampaignConstants.ELEMENT_COLLATERAL_NAME_REQUIRED %>");
      alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
              CampaignConstants.MSG_COLLATERAL_NAME_REQUIRED)) %>");
      <%= CampaignConstants.ELEMENT_COLLATERAL_NAME %>.focus();
   }
```

```
if (parent.get("<%= CampaignConstants.ELEMENT_COLLATERAL_NAME_INVALID %>", false)) {
   parent.remove("<%= CampaignConstants.ELEMENT_COLLATERAL_NAME_INVALID %>");
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
           CampaignConstants.MSG_PLEASE_ENTER_AN_ALPHANUMERIC_NAME)) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_NAME %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_NAME %>.focus();
}

if (parent.get("collateralNameTooLong", false)) {
   parent.remove("collateralNameTooLong");
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
           "collateralNameTooLong")) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_NAME %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_NAME %>.focus();
}

if (parent.get("collateralCustomField1TooLong", false)) {
   parent.remove("collateralCustomField1TooLong");
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
           "collateralCustomField1TooLong")) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_FIELD1 %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_FIELD1 %>.focus();
}

if (parent.get("collateralCustomField2TooLong", false)) {
   parent.remove("collateralCustomField2TooLong");
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
           "collateralCustomField2TooLong")) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_FIELD2 %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_FIELD2 %>.focus();
}

if (parent.get("<%= CampaignConstants.ELEMENT_COLLATERAL_EXISTS %>", false)) {
   parent.remove("<%= CampaignConstants.ELEMENT_COLLATERAL_EXISTS %>");
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
           CampaignConstants.MSG_COLLATERAL_EXISTS)) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_NAME %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_NAME %>.focus();
}

if (parent.get("collateralLocationTooLong", false)) {
   parent.remove("collateralLocationTooLong");
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
           "collateralLocationTooLong")) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_LOCATION %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_LOCATION %>.focus();
}

if (parent.get("collateralMarketingTextTooLong", false)) {
   parent.remove("collateralMarketingTextTooLong");
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
           "collateralMarketingTextTooLong")) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %>.focus();
}

if (parent.get("collateralDescription1TooLong", false)) {
   parent.remove("collateralDescription1TooLong");
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
           "collateralDescription1TooLong")) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %>.focus();
}

if (parent.get("collateralDescription2TooLong", false)) {
   parent.remove("collateralDescription2TooLong");
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
           "collateralDescription2TooLong")) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %>.focus();
}
```

j. Update the validatePanelData function with the following validation checks from the description panel:

Note: This code sample below was created by copying code from the CollateralDescriptionPanel.jsp file, and merging it with code originally located in the CollateralGeneralPanel.jsp. No new code was required.

```
if (!isValidUTF8length(<%=
           CampaignConstants.ELEMENT_COLLATERAL_LOCATION %>.value, 254)) {
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
                       "collateralLocationTooLong")) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_LOCATION %>.select();
```

```
   <%= CampaignConstants.ELEMENT_COLLATERAL_LOCATION %>.focus();
   return false;
}
if (!isValidUTF8length(<%=
            CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %>.value, 4000)) {
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
                       "collateralMarketingTextTooLong")) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %>.focus();
   return false;
}
if (!isValidUTF8length(<%=
            CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %>.value, 254)) {
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
                       "collateralDescription1TooLong")) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %>.focus();
   return false;
}
if (!isValidUTF8length(<%=
            CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %>.value, 254)) {
   alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
                       "collateralDescription2TooLong")) %>");
   <%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %>.select();
   <%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %>.focus();
   return false;
}
```

k. Update the savePanelData function with the following code from the description panel:

   Note: This code sample below was created by copying code from the CollateralDescriptionPanel.jsp file, and merging it with code originally located in the CollateralGeneralPanel.jsp. No new code was required.

```
o.<%= CampaignConstants.ELEMENT_COLLATERAL_LOCATION %> =
             <%= CampaignConstants.ELEMENT_COLLATERAL_LOCATION %>.value;
o.<%= CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %> =
             <%= CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %>.value;
o.<%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %> =
             <%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %>.value;
o.<%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %> =
             <%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %>.value;
```

l. Update the gotoSearchSkuDialog function. This function defines the action to take if the user clicks **Find** to find and identify a SKU of a product to which the ad copy links. Before calling the page to handle the Find interaction, this function defines the page to which the WebSphere Commerce Accelerator returns after the user finished using the Find dialog. This must be updated to reflect the custom page. To change this, update the following code:

```
top.setReturningPanel("collateralGeneralPanel");
```

   Update it so that it references the customized panel name, as defined in the customized wizard and notebook XML files. These will be created in a later step. The updated code should look like the following:

```
top.setReturningPanel("ConsolidatedCollateralNavigation");
```

m. Make the same change in the gotoBrowseSkuDialog method. Update the following code:

```
top.setReturningPanel("collateralGeneralPanel");
```

   Update it so that it references the customized panel name, as defined in the customized wizard and notebook XML files. The updated code should look like the following:

```
top.setReturningPanel("ConsolidatedCollateralNavigation");
```

n. Combine the form elements in some logical manner. A form design is illustrated below:

Note: This code sample below was created by copying code from the CollateralDescriptionPanel.jsp file, and merging it with code originally located in the CollateralGeneralPanel.jsp. No new code was required.

```
<FORM NAME="generalForm">

<input name="<%= CampaignConstants.ELEMENT_COLLATERAL_URL %>" type="hidden">
<input name="<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND %>" type="hidden">

<P><%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_NAME_PROMPT) %><BR>
<INPUT NAME="<%= CampaignConstants.ELEMENT_COLLATERAL_NAME %>"
              TYPE="TEXT" size=30 maxlength=30>

<P><%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_LOCATION_PROMPT) %><BR>
<INPUT NAME="<%= CampaignConstants.ELEMENT_COLLATERAL_LOCATION %>"
              TYPE="TEXT" size=30 maxlength=254>

<P><%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_TYPE_PROMPT) %><BR>
<select name="<%= CampaignConstants.ELEMENT_COLLATERAL_TYPE %>">
<% for (int i=1; i<=CampaignConstants.COLLATERAL_TYPE_COUNT; i++) { %>
   <OPTION value="<%= i %>">
      <%= (String)campaignsRB.get(CampaignConstants.ELEMENT_COLLATERAL_TYPE + i) %>
   </OPTION>
<% } %>
</select>

<P><INPUT NAME="<%= CampaignConstants.ELEMENT_COLLATERAL_URL_TYPE %>" TYPE="RADIO"
              VALUE="true" ONCLICK="showDivisions()" CHECKED>
<%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_URL_COMMAND_ACTION_PROMPT) %>
<div id="commandUrlDiv" style="display: none; margin-left: 20">
<table border="0" cellspacing="3" cellpadding="3">
   <tr><td>
      <%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_URL_COMMAND_PROMPT) %>
      <br>
      <select name="<%= CampaignConstants.ELEMENT_COLLATERAL_URL_COMMAND_TYPE %>"
              onChange="javascript:updateCollateralUrl(this);">
      <% for (int i=0; i<CampaignConstants.collateralUrlArray.length; i++) { %>
      <OPTION value="<%= CampaignConstants.collateralUrlArray[i] %>">
         <%= campaignsRB.get("collateralUrl" + CampaignConstants.collateralUrlArray[i]) %>
      </OPTION>
      <% } %>
  </select>
   </td></tr>
   <tr><td>
      <div id="commandProductDiv" style="display: none;">
      <%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_URL_PRODUCT_PROMPT) %>
      <br>
      <table border="0" cellpadding="0" cellspacing="0">
         <tr>
         <td valign="top"><INPUT NAME="urlCommandProduct" TYPE="TEXT" size="30"
                        maxlength="254" readonly></td>
         <td valign="top">
            <BUTTON type="BUTTON" name="searchButton" value="searchSKU" CLASS="enabled"
                        onClick="gotoSearchSkuDialog();">
               <%= campaignsRB.get(CampaignConstants.MSG_BUTTON_FIND_ELLIPSIS) %>
            </BUTTON>
            <BUTTON type="BUTTON" name="browseButton" value="browseSKU" CLASS="enabled"
                         onClick="gotoBrowseSkuDialog();">
               <%= campaignsRB.get(CampaignConstants.MSG_BUTTON_BROWSE_ELLIPSIS) %>
            </BUTTON>
         </td>
         </tr>
      </table>
      </div>
      <div id="commandCategoryDiv" style="display: none;">
      <%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_URL_CATEGORY_PROMPT) %>
      <br>
      <table border="0" cellpadding="0" cellspacing="0">
         <tr>
         <td valign="top"><INPUT NAME="urlCommandCategory" TYPE="TEXT" size="30"
                        maxlength="254" readonly></td>
         <td valign="top">
            <BUTTON type="BUTTON" name="listCategoryButton" value="listCategory"
                        CLASS="enabled" onClick="gotoSearchSkuDialog();">
               <%= campaignsRB.get(CampaignConstants.MSG_BUTTON_LIST_ELLIPSIS) %>
            </BUTTON>
         </td>
         </tr>
      </table>
      </div>
      <div id="commandCouponDiv" style="display: none;">
      <%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_URL_COUPON_PROMPT) %>
      <br>
      <table border="0" cellpadding="0" cellspacing="0">
         <tr>
```

```
            <td valign="top"><INPUT NAME="urlCommandCoupon" TYPE="TEXT" size="30"
                                maxlength="254" readonly></td>
            <td valign="top">
               <BUTTON type="BUTTON" name="listCouponButton" value="listCoupon"
                                CLASS="enabled" onClick="gotoSearchSkuDialog();">
                  <%= campaignsRB.get(CampaignConstants.MSG_BUTTON_LIST_ELLIPSIS) %>
               </BUTTON>
            </td>
            </tr>
         </table>
         </div>
      </td></tr>
   </table>
   </div>
   <INPUT NAME="<%= CampaignConstants.ELEMENT_COLLATERAL_URL_TYPE %>" TYPE="RADIO"
                                VALUE="false" ONCLICK="showDivisions()">
   <%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_URL_CUSTOM_ACTION_PROMPT) %>
   <div id="customUrlDiv" style="display: none; margin-left: 20">
   <table border="0" cellspacing="3" cellpadding="3">
      <tr><td>
         <%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_URL_PROMPT) %>
         <br>
         <input name="<%= CampaignConstants.ELEMENT_COLLATERAL_URL_CUSTOM %>"
                                type="text" size="60" maxlength="254">
      </td></tr>
   </table>
   </div>

   <P><%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_MARKETING_TEXT_PROMPT) %><BR>
   <TEXTAREA NAME="<%= CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %>" ROWS="4"
         COLS="50" WRAP=physical onKeyDown="limitTextArea(
               this.form.<%= CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %>, 4000);"
         onKeyUp="limitTextArea(this.form.<%=
               CampaignConstants.ELEMENT_COLLATERAL_MARKETING_TEXT %>, 4000);">
   </TEXTAREA>

   <P><%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_FIELD1_PROMPT) %><BR>
   <TEXTAREA NAME="<%= CampaignConstants.ELEMENT_COLLATERAL_FIELD1 %>" ROWS="4"
         COLS="50" WRAP=physical onKeyDown="limitTextArea(
               this.form.<%= CampaignConstants.ELEMENT_COLLATERAL_FIELD1 %>, 254);"
         onKeyUp="limitTextArea(this.form.<%=
               CampaignConstants.ELEMENT_COLLATERAL_FIELD1 %>, 254);">
   </TEXTAREA>

   <P><%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_FIELD2_PROMPT) %><BR>
   <TEXTAREA NAME="<%= CampaignConstants.ELEMENT_COLLATERAL_FIELD2 %>" ROWS="4"
               COLS="50" WRAP=physical
         onKeyDown="limitTextArea(this.form.<%=
               CampaignConstants.ELEMENT_COLLATERAL_FIELD2 %>, 254);"
         onKeyUp="limitTextArea(this.form.<%=
               CampaignConstants.ELEMENT_COLLATERAL_FIELD2 %>, 254);">
   </TEXTAREA>

   <P><%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_DESC_FIELD1_PROMPT) %><BR>
   <TEXTAREA NAME="<%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %>" ROWS="4"
               COLS="50" WRAP=physical
         onKeyDown="limitTextArea(this.form.<%=
               CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %>, 254);"
         onKeyUp="limitTextArea(this.form.<%=
               CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD1 %>, 254);">
   </TEXTAREA>

   <P><%= campaignsRB.get(CampaignConstants.MSG_COLLATERAL_DESC_FIELD2_PROMPT) %><BR>
   <TEXTAREA NAME="<%= CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %>" ROWS="4"
               COLS="50" WRAP=physical
         onKeyDown="limitTextArea(this.form.<%=
               CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %>, 254);"
         onKeyUp="limitTextArea(this.form.<%=
               CampaignConstants.ELEMENT_COLLATERAL_DESC_FIELD2 %>, 254);">
   </TEXTAREA>

   </FORM>
```

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

2. Register the new page in the VIEWREG database table. This table governs the relationships between views, and the JSP files they launch. Because this customization creates a new view, you must make a corresponding entry in this table. To update the VIEWREG table, use the following SQL statement:

```
insert into VIEWREG
   (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME, CLASSNAME,
    PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE, INTERNAL)
  values
   ('ConsolidatedCollateralPanelView', -1, Store_ID,
```

```
'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commerce.tools.command.ToolsForwardViewCommandImpl',
'docname=tools/custom/ConsolidatedCollateralPanel.jsp',
'A custom view.', 0,  NULL, 0)
```

3. Create a new wizard XML file. To create this file, do the following:
   a. Create a new directory. For the purposes of this scenario, name the directory, */myCustomXML*/tools/campaigns/, where *myCustomXML* represents some custom directory name.

      Note: If you completed "Adding an action button to the Campaigns list page" on page 3, this directory will already exist.
   b. Copy the */WCDE_installdir*/xml/tools/campaigns/CollateralWizard.xml file, and move the file to the */myCustomXML*/tools/campaigns/ directory.
   c. Open the new CollateralWizard.xml file in an editor.
   d. Edit the file so that it matches the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE wizard SYSTEM "../common/WizardPanels.dtd">
<!--
 *=====================================================================
 * Some descriptive comments as appropriate
 *
 *=====================================================================
-->

<wizard resourceBundle="campaigns.campaignsRB"
    windowTitle="collateralWizardTitle"
    finishConfirmation=""
    cancelConfirmation="collateralWizardCancelConfirmation"
   tocBackgroundImage="/wcs/images/tools/toc/W_market.jpg"
    finishURL="CampaignCollateralSave" >

    <panel name="ConsolidatedCollateralNavigation"
        url="ConsolidatedCollateralPanelView"
        hasFinish="YES"
        helpKey="MC.campaigns.ConsolidatedCollateralPanel.Help" />

    <databean name="collateral"
        class="com.ibm.commerce.tools.campaigns.CampaignCollateralDataBean" />

    <jsFile src="/wcs/javascript/tools/campaigns/Collateral.js" />

</wizard>
```

4. Create a new notebook XML file. To create this file, do the following:
   a. Copy the */WCDE_installdir*/xml/tools/campaigns/CollateralNotebook.xml file, and move the file to the */myCustomXML*/tools/campaigns/ directory.
   b. Open the new CollateralNotebook.xml file in an editor.
   c. Edit the file so that it matches the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE notebook SYSTEM "../common/NotebookPanels.dtd">
<!--
 *=====================================================================
 * Some descriptive comments as appropriate
 *
 *=====================================================================
-->

<notebook resourceBundle="campaigns.campaignsRB"
      windowTitle="collateralNotebookTitle"
      finishConfirmation=""
      cancelConfirmation="collateralNotebookCancelConfirmation"
      tocBackgroundImage="/wcs/images/tools/toc/NB_market.jpg"
      finishURL="CampaignCollateralSave" >

   <panel name="ConsolidatedCollateralNavigation"
         url="CosolidatedCollateralPanelView"
         helpKey="MC.campaigns.ConsolidatedCollateralPanel.Help" />
```

```
    <databean name="collateral"
          class="com.ibm.commerce.tools.campaigns.CampaignCollateralDataBean" />

    <jsFile src="/wcs/javascript/tools/common/Util.js" />
    <jsFile src="/wcs/javascript/tools/campaigns/Collateral.js" />

</notebook>
```

5. Update the Campaigns properties file with the new required strings. To make the updates, do the following:
   a. In the WebSphere Commerce development environment, navigate to the WebSphereCommerceServerExtensionsLogic project.
   b. Navigate to the src folder.

      Note: If you completed "Adding an action button to the Campaigns list page" on page 3, the custom properties file already exists. Double-click the file to open it, and skip to step 5i.
   c. Right-click on the src folder, and select **New**, and then **Package**.
   d. In the **Name Field** on the New Java Package dialog, enter a unique name. For the purposes of this scenario, enter `com.`*`myCompany`*`.campaigns.properties`, where *myCompany* represents some custom directory name. Packages created by IBM, and included in WebSphere Commerce follow a naming convention which begins, "com.ibm..."
   e. Click **Finish** to create the package.
   f. Right-click on the com.*myCompany*.campaigns.properties folder, and select **New**, and then **Other**. In the New dialog, click **Simple**, **File**, and then **Next**.
   g. In the **File name** field, enter a unique name. For the purposes of this scenario, enter `CampaignsRB_`*`locale`*`.properties`, where *locale* represents the locale from which your business users will access the WebSphere Commerce Accelerator.
   h. Click **Finish** to create the file and open it in an editor.
   i. Create a "Collateral Notebook" section for the file. This section contains custom strings for your customized notebook. The only required string at this time defines the title for the new page:

      ```
      #
      # Collateral Notebook
      #

      ConsolidatedCollateralPanel=Consolidated Ad Copy
      ConsolidatedCollateralNavigation=Ad Copy
      ```
   j. Create a "Buttons" section for the file. In addition to defining the consolidated page, we must customize the Collateral List page so that it can launch the new notebook. In the new section, insert labels for the new buttons. This should look similar to the following sample:

      ```
      #
      # Buttons
      #

      NewChangeCollateral=Change Collateral
      NewCreateCollateral=Create Collateral
      ```
   k. Save the file, but do not close the development environment.

   This addresses the label in the language for the locale you specified when 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 CampaignsRB.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.

6. Update the Collateral List definition XML file to add two new buttons, Create and Update, which lead to the consolidated panel. Again, to protect your customized data, it must be created in a safe place, separate from the WebSphere Commerce assets. This procedure creates a new Collateral List definition file, in a new folder. This is not done within the development environment. To add the button, do the following:
   a. Create a new directory. For the purposes of this scenario, name the directory, */myCustomXML*/tools/campaigns/, where *myCustomXML* represents some custom directory name.

      Note: If you completed "Adding an action button to the Campaigns list page" on page 3, this folder will already exist.
   b. Copy */WCDE_installdir*/xml/tools/campaigns/CollateralList.xml to the new */myCustomXML*/tools/campaigns/ directory. Do not change the file name.
   c. Open the CollateralList.xml file in an editor.
   d. Scroll down to the code segment about buttons. Add the following code:

```
<menu name="NewChangeCollateral"
   action="basefrm.collateralPropertiesChanged()"
   selection="single" />
<menu name="NewCreateCollateral"
   action="basefrm.newCollateralChanged()"
   selection="none" />
```

      The actions specified here must match the name used for the function in the Collateral List JSP file, which follow in step 8.

This scenario does not include removing the existing **Create** and **Change** buttons, but you are free to make this change if you wish.

   e. Save the file.
7. Since we have created custom XML files to define a wizard and a notebook, and customized the existing version of the CollateralList JSP, and created a custom properties file, you must customize the appropriate resources.xml file so that the customized user interface displays as expected. This is not done within the development environment. To make this update, do the following:
   a. Copy the */WCDE_installdir*/xml/tools/campaigns/resources.xml file, and move the file to the */myCustomXML*/tools/campaigns/ directory.

      Note: If you completed "Adding an action button to the Campaigns list page" on page 3, this folder will already contain a custom resources.xml. You should make the changes to the existing file.
   b. Open the new resources.xml file in an editor.
   c. Scroll down to the ″Resource bundle″ section of the file, and update the code so that it matches the following sample:

```
<resourceBundle name="campaignsRB">
    <bundle>com.ibm.commerce.tools.campaigns.properties.CampaignsRB</bundle>
    <bundle>com.myCompany.campaigns.properties.CampaignsRB</bundle>
</resourceBundle>
```

where *myCompany* represents your custom directory name.

d. Scroll down to the ″XML file mappings″ section of the file, and update the following entries:

- Update the collateral wizard entry so that it matches the following sample:

```
<resourceXML name="CampaignCollateralWizard"
        file="/myCustomXML/tools/campaigns/CollateralWizard.xml" />
```

  where *myCustomXML* represents your custom directory name.

  Note: The file name specified here must match the name used when the notebook XML file was created in step 2.

- Update the collateral notebook entry so that it matches the following sample:

```
<resourceXML name="ConsolidatedCollateralNotebook"
    file="/myCustomXML/tools/campaigns/CollateralNotebook.xml">
```

  where *myCustomXML* represents your custom directory name.

  Note: The file name specified here must match the name used when the notebook XML file was created in step 2.

- Update the collateral list entry so that it matches the following sample:

```
<resourceXML name="CampaignCollateralList"
    file="/myCustomXML/tools/campaigns/CollateralList.xml">
```

  where *myCustomXML* represents your custom directory name.

  Note: The file name specified here must match the name used when the list XML file was updated in step 5.

e. Save the file.

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 an 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 your custom directory name.

   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. Update the CollateralList.jsp with a new functions to call the new wizard, and the new notebook. To update the JSP, do the following:
   a. In the WebSphere Commerce development environment, navigate to the CommerceAccelerator/Web Content/tools/campaigns folder.

b. Right-click on the CollateralList.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 an editor.
f. Add the following function to call the new wizard (lines may be split here for presentation purposes):

```
function newCollateralChanged () {
  var url = "/webapp/wcs/tools/servlet/WizardView?XMLFile=
          campaigns.ConsolidatedCollateralWizard";
  var panelTitle =
          "<%=UIUtil.toJavaScript((String)campaignsRB.get("NewCreateCollateral"))%>";
  if (top.setContent) {
  top.setContent(panelTitle, url, true); }
 else {
  parent.location.replace(url);
 }
}
```

g. Add the following function to call the new notebook (lines may be split here for presentation purposes):

```
function collateralPropertiesChanged () {
    var collateralId = -1;
    if (arguments.length > 0) {
       collateralId = arguments[0];
    }
    else {
       var checked = parent.getChecked();
       if (checked.length > 0) {
          if (getListEditableFlag(checked[0]) == "Y") {
             collateralId = getListCollateralId(checked[0]);
          }
          else {
             alertDialog("<%= UIUtil.toJavaScript((String)campaignsRB.get(
                         "listEntryCannotBeModified")) %>");
          }
       }
    }
 var url = "/webapp/wcs/tools/servlet/NotebookView?XMLFile=
          campaigns.ConsolidatedCollateralNotebook";
 var panelTitle =
          "<%=UIUtil.toJavaScript((String)campaignsRB.get("NewCreateCollateral"))%>";
    if (collateralId != -1) {
     url += "&collateralID=" + collateralId;
     panelTitle =
          "<%=UIUtil.toJavaScript((String)campaignsRB.get("NewChangeCollateral"))%>";
    }
    if (top.setContent) {
       top.setContent(panelTitle, url, true);
    }
    else {
       parent.location.replace(url);
    }
 }
```

h. Also, because the custom file is in a different location, you have to update the following path in the JSP to reflect the original location. Search for the following include statement:

```
<%@ include file="common.jsp" %>
```

and change it to:

```
<%@ include file="../campaigns/common.jsp" %>
```

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

10. Register this new page in the VIEWREG database table. This table governs the relationships between views, and the JSP files they launch. Because this customization creates a new view, you must make a corresponding entry in this table. To update the VIEWREG table, use the following SQL statement:

```
update VIEWREG set PROPERTIES = 'docname=tools/custom/CollateralList.jsp'
   where VIEWNAME='CampaignCollateralListView'
```

11. 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 **Marketing** menu, select **Ad Copy**. Two new buttons should display on this page.
    d. Click the new **Create Collateral** button. This should launch the custom Ad Copy notebook. Enter a name for the new ad copy, and click **Finish** to create the ad copy.
    e. Once you are returned to the ad copy list, select your new ad copy, and click the new **Change Collateral** button. This should load the new consolidated page with the details of the ad copy you just created.
    f. You should also test the **Find** and **Browse** buttons to ensure that they function properly.
    g. If all of these steps work, 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 fails, you will have to determine the cause of the error, and debug.
12. Export the updated assets:
    a. Right-click the CollateralList.jsp file and select **Export**. The Export Wizard opens.

       In the Export wizard, do the following:
       1) Select **File system** and click **Next**.
       2) The CollateralList.jsp is selected.
       3) Ensure that you also select ConsolidatedCollateralPanel.jsp.
       4) Navigate to the WebSphereCommerceServerExtensionsLogic/src/com/*myCompany*/campaigns/properties folder, and select the myCampaignsRB_*locale*.properties file.
       5) Select **Create directory structure for selected files**.
       6) In the **To directory** field, enter a temporary directory into which these resources will be placed. For example, enter `C:\ExportTemp\StoreAssets`
       7) Click **Finish**.
       8) If prompted, create the specified directory.
13. 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 is 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 12, above, into the following directories:

CollateralList.jsp
Copy to
WC_*instance_name*.ear/CommerceAccelerator.war/tools/custom

ConsolidatedCollateralPanel.jsp
Copy to
WC_*instance_name*.ear/CommerceAccelerator.war/tools/custom

myCampaignsRB_*locale***.properties**
Copy to WC_*instance_name*.ear/properties/com/*myCompany*/campaigns/properties

d. Transfer the following files to your production environment:

/myCustomXML/**tools/campaigns/CollateralList.xml**
Copy to WC_*installdir*/xml/tools/custom

/myCustomXML/**tools/campaigns/CollateralWizard.xml**
Copy to WC_*installdir*/xml/tools/custom

/myCustomXML/**tools/campaigns/CollateralNotebook.xml**
Copy to WC_*installdir*/xml/tools/custom

/WCDE_installdir/**xml/tools/campaigns/resources.xml**
Copy to WC_*installdir*/xml/tools/campaigns

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 campaign component is based upon the following assets.

JSP assets

The ad copy 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/ 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 ad copy JSP files.

Table 4. Ad copy JSP Assets

Page	JSP	Properties
List	CollateralList.jsp	*/WCDE_installdir*/properties/runtime/com/ibm/commerce/tools/campaigns/properties/CampaignsRB_*locale*.properties
General Definition	CollateralGeneralPanel.jsp	
Description Definition	CollateralDescriptionPanel.jsp	
Summary	CollateralPreviewPanel.jsp	

XML assets

The ad copy 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 */WCDE_installdir*/xml/tools/campaigns 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 ad copy XML files that may require updates.

Table 5. Ad copy XML files

XML filename	Description
CollateralDeletedDialog.xml	Defines the framework and content for the Collateral Deleted dialog.
CollateralList.xml	Defines the framework for the Collateral List page. This includes navigation, and action buttons, but the content in the primary frame originates in the CollateralList.jsp file.
CollateralNotebook.xml	Defines the framework for the Collateral Notebook. This includes navigation, and action buttons, but the content in the primary frame originates in the CollateralGeneralPanel.jsp and CollateralDescriptionPanel.jsp files.
CollateralPreviewDialog.xml	Defines the framework and content for the Collateral Summary dialog.
CollateralWizard.xml	Defines the framework for the Collateral Wizard. This includes navigation, and action buttons, but the content in the primary frame originates in the CollateralGeneralPanel.jsp and CollateralDescriptionPanel.jsp files.

Database assets

The ad copy component makes use of data stored in the database table described in the following table.

Table 6. Ad copy database tables

Table name	Column name	Additional information
COLLATERAL	COLLATERAL_ID	System generated.
	COLLTYPE_ID	Reference to COLLTYPE table.
	FIELD1	Available for custom information.
	FIELD2	Available for custom information.
	NAME	Entered on the General page.
	STOREENT_ID	Reference to STOREENT table.
	URL	Entered on the General page.
COLLDESC	COLLATERAL_ID	Reference to COLLATERAL table.
	FIELD1	Available for custom information.
	FIELD2	Available for custom information.
	LANGUAGE_ID	Reference to LANGUAGE table.
	LOCATION	Fully qualified path information to the file.
	MARKETINGTEXT	Entered on the Description page.
COLLTYPE	COLLTYPE_ID	System generated.
	NAME	Populated by bootstrap data.

Notices

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 that IBM product, program, or service may be used. Any functionally equivalent 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 subject matter described in this document. The furnishing of this document does not give 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-chome, Minato-ku
Tokyo 106, 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 independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Canada Ltd.
Office of the Lab Director
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 document 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.

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:

400	AIX	IBM
iSeries	OS/400	WebSphere

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

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

Printed in USA