BEA WebLogic Portal™

Developing Campaign Infrastructure

Version 4.1
Document Date: December 2001

Guide to Developing Campaign Infrastructure

Document Edition	Date	Software Version
4.1	December 2001	BEA WebLogic Portal 4.0, Service Pack 1
4.0	October 2001	BEA WebLogic Portal 4.0

Contents

About This Document

This document describes setting up infrastructure for the features that the Campaign Service uses. It includes the following topics:

- Chapter 1, “Roadmap for Developing Campaign Infrastructure,” which provides an overview of the concepts and tasks involved in setting up campaign infrastructure.
- Chapter 2, “Setting Up Ads for Campaigns,” which discusses loading advertising documents (ads) into a content management system and using attributes to describe the documents.
- Chapter 3, “Setting Up JSP Tags and Scriptlets for Campaigns,” which provides instructions for creating JSP tags to support campaign features.
- Chapter 4, “Setting Up and Sending E-mail for Campaigns,” which provides instructions for setting up and using the default Mail Service to support campaigns.
- Chapter 5, “Campaign Manager Database Schema,” which the describes the database schema for the Campaign services. Understanding this schema will be helpful to those who may be customizing or extending the technologies provided in the product.

What You Need to Know

This document is intended for Commerce Business Engineers (CBE). It assumes that you are familiar with developing Java Server Pages (JSP) and creating Java scriptlets and JSP tags. In addition, it assumes that you are familiar with maintaining your content management system.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the “e-docs” Product Documentation page at http://e-docs.bea.com.

How to Print the Document

You can print a copy of this document from a Web browser, one file at a time, by using the File—>Print option on your Web browser.

A PDF version of this document is available on the WebLogic Portal documentation Home page on the e-docs Web site. A PDF version of this document is also available in the documentation kit on the product CD. Or you can download the documentation kit from the WebLogic Portal portion of the BEA Download site. You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Portal documentation Home page, click the PDF files button and select the document you want to print.

If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at http://www.adobe.com/.

Related Information

For information on setting up related WebLogic Portal services, refer to the following documents:

- *Guide to Events and Behavior Tracking*
- *Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline*

- *Guide to Building a Product Catalog*
- *Guide to Managing Purchases and Processing Orders*
- *Guide to Registering Customers and Managing Customer Services*
- *Guide to Developing and Managing Portals*
- *Guide to Building Personalized Applications*

To see an example implementation of a campaign, refer to the *JSP Commerce and Campaign Tour*.

Contact Us!

Your feedback on the BEA WebLogic Portal documentation is important to us. Send us e-mail at **docsupport@bea.com** if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the WebLogic Portal documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Portal **Product Version: 4.0** release.

If you have any questions about this version of BEA WebLogic Portal, or if you have problems installing and running BEA WebLogic Portal, contact BEA Customer Support through BEA WebSUPPORT at **www.bea.com**. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Examples*: LPT1 SIGNON OR

Convention	Item
`{ }`	Indicates a set of choices in a syntax line. The braces themselves should never be typed.

`[ ]`	Indicates optional items in a syntax line. The brackets themselves should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

`|`	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

`...`	Indicates one of the following in a command line:
	■ That an argument can be repeated several times in a command line
	■ That the statement omits additional optional arguments
	■ That you can enter additional parameters, values, or other information
	The ellipsis itself should never be typed.
	Example:
	`buildobjclient [-v] [-o name ] [-f` *`file-list`*`]...`
	`[-l` *`file-list`*`]...`

. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Roadmap for Developing Campaign Infrastructure

Developing campaign infrastructure is a collaborative effort between a Business Analyst (BA) and a Business Engineer (BE). A BA develops the strategy and goals for individual campaigns, and uses the BEA E-Business Control Center to run and evaluate them. A BE develops the infrastructure to support campaigns and modifies the infrastructure as individual campaigns require.

This topic provides an overview of developing campaign infrastructure from the BE perspective. It includes the following sections:

- What Is a Campaign?
- How Data Flows in a Campaign
- Workflow for Developing Campaign Infrastructure

What Is a Campaign?

A campaign coordinates several WebLogic Portal services to create and track marketing goals on an e-commerce Web site. For example, your Marketing organization can use campaigns to sell 100 ACME saws during the month of June. To reach this goal, Marketing can target advertising, e-mail, and discounted product pricing to customers who match a set of criteria, such as customers who have previously purchased ACME hardware from your site.

Note: Campaigns cannot be used with anonymous users.

Campaigns coordinate the following services:

- Events and Behavior Tracking Services, which identify how a customer interacts with your site. By default WebLogic Portal tracks only a specific set of customer interactions (events), but you can add to this set by customizing the Event Service.
- Customer Segments, which categorize customers based on information in a customer's profile and other dynamic data. Each customer must create a profile to log in to your site. The profile includes information that the customer provides, such as shipping addresses, and information that WebLogic Portal provides, such as number of visits and total value of products the customer has purchased on the site. A BA creates customer segments in the E-Business Control Center.
- Scenarios, which trigger actions if a specific event occurs or if a specific customer matches a customer segment. A BA creates scenarios in the E-Business Control Center.

  Scenarios can engage any of the following services:

  - Ad Placeholders, which query the content management system for an ad and display the query results on the Web site. For example, if a customer logs in and the customer's profile matches the SailingEnthusiast customer segment, then a scenario displays an ad for sailboats in the Web site's top banner.
  - The E-mail Service, which uses a JSP to generate e-mail and provides a utility for sending the e-mail in batches. Because the Mail Service uses a JSP to generate e-mail, you can use JSP tags to personalize the e-mail.

- Discounts, which offer reduced prices for specific products or product categories. A BA creates discounts in the E-Business Control Center.

How Data Flows in a Campaign

In a typical campaign, data flows as follows (see Figure 1-1):

1. An event is generated by a trigger. For example, a customer logs in.

   In general, an event is a notification that something has happened in a computer program. For more information about the Event Service, refer to the *Guide to Events and Behavior Tracking*.

Note: Campaigns cannot be used with anonymous users. Campaigns require a user ID that has two characteristics: the ID must be associated with a user profile, and that user profile must be saved (persisted). However, the anoymous profile for a user who is not logged in is a runtime profile (not saved), and not associated with a user ID.

Personalization features such as <pz:div> and <pz:contentSelector> JSP tags do work for anonymous users. This is because these features can use a runtime profile without a user ID.

2. The Event Service creates an event object to encapsulate information about the event. If a customer triggers the event, the event object includes the customer ID.

3. The Event Service notifies all Event Listeners that it has detected an event. The event listener for the Campaign Service determines whether the event object is one of the following types:

   - **Session**: The start time, end time, and if executed, the login time of the customer’s session.
   - **Registration:** The customer registers on the e-commerce site.
   - **Product**: The customer is presented with a product or clicks (selects) the presented product.
   - **Content**: The customer is presented some content, such as an ad, or clicks (selects) the presented content.

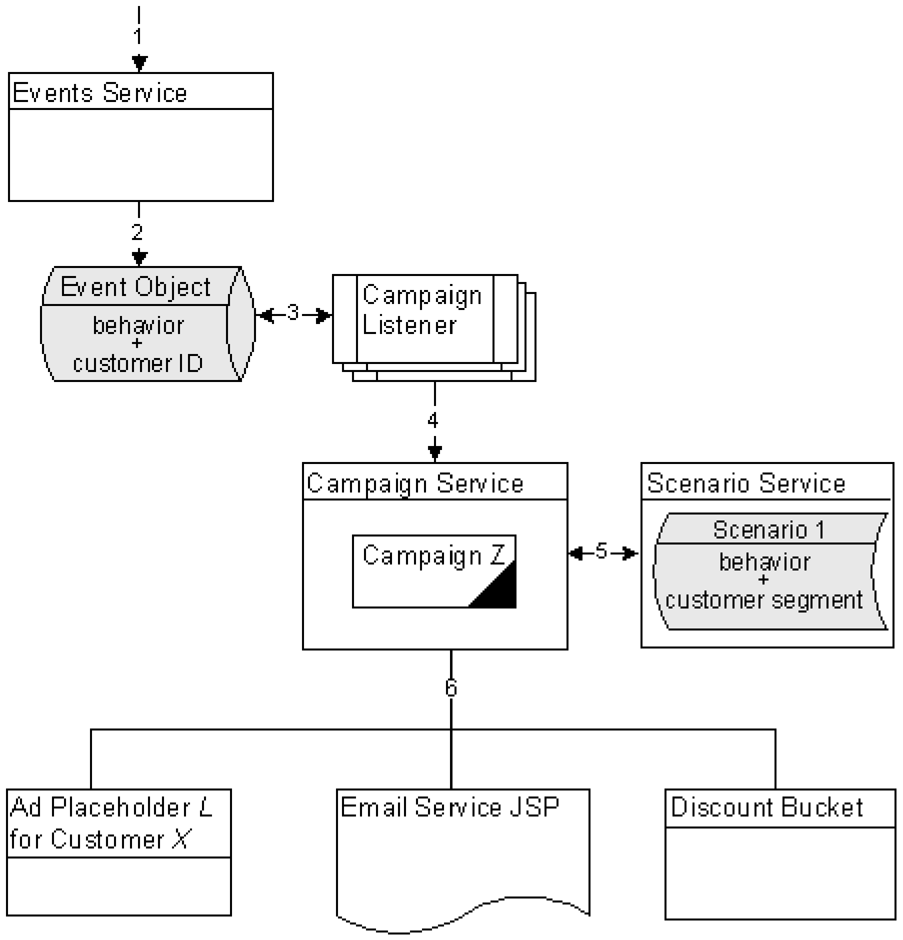
- **Cart**: An item is added, removed, or updated to the customer's shopping cart.
- **Buy**: The customer completes the purchase of one or more items.
- **Campaign**: The events generated within the context of a campaign.

4. If the event is one of the types described in the previous step, the Campaign Listener notifies the Campaign Service.
5. For each active campaign, the Campaign Service attempts to match the event with a scenario. For example, customer Pat Gomes is a Gold Customer who added a handsaw to the shopping cart. Scenario 1 in Campaign Z specifies actions for Gold Customers who add handsaws to the shopping cart.
6. For any scenario that matches the event, the Scenario Service finds an action to initiate. Then the Campaign Service initiates the action. For example, If a Gold Customer adds a handsaw to the shopping cart, run a query for documents that advertise drills and display the results in the shopping cart JSP.

Ad Placeholders, the E-mail Service, and the Discount Service operate independently of the Campaign Service. The following sections describe how these services process the data that a campaign gives to them:

- How Placeholders Select and Display Ads for Campaigns
- How Campaigns Use the Mail Service
- How Campaigns Offer Discounts

Figure 1-1 Data Flow in a Campaign



How Placeholders Select and Display Ads for Campaigns

Placeholders use the following process to select and display ads in a given JSP (see Figure 1-2):

1. As part of carrying out a campaign action, the Campaign Service adds queries to the placeholder.
2. When a user requests a JSP that contains a placeholder, if the ad placeholder contains more than one ad query, the Ad Service calls the Ad Conflict Resolver to select an ad query.

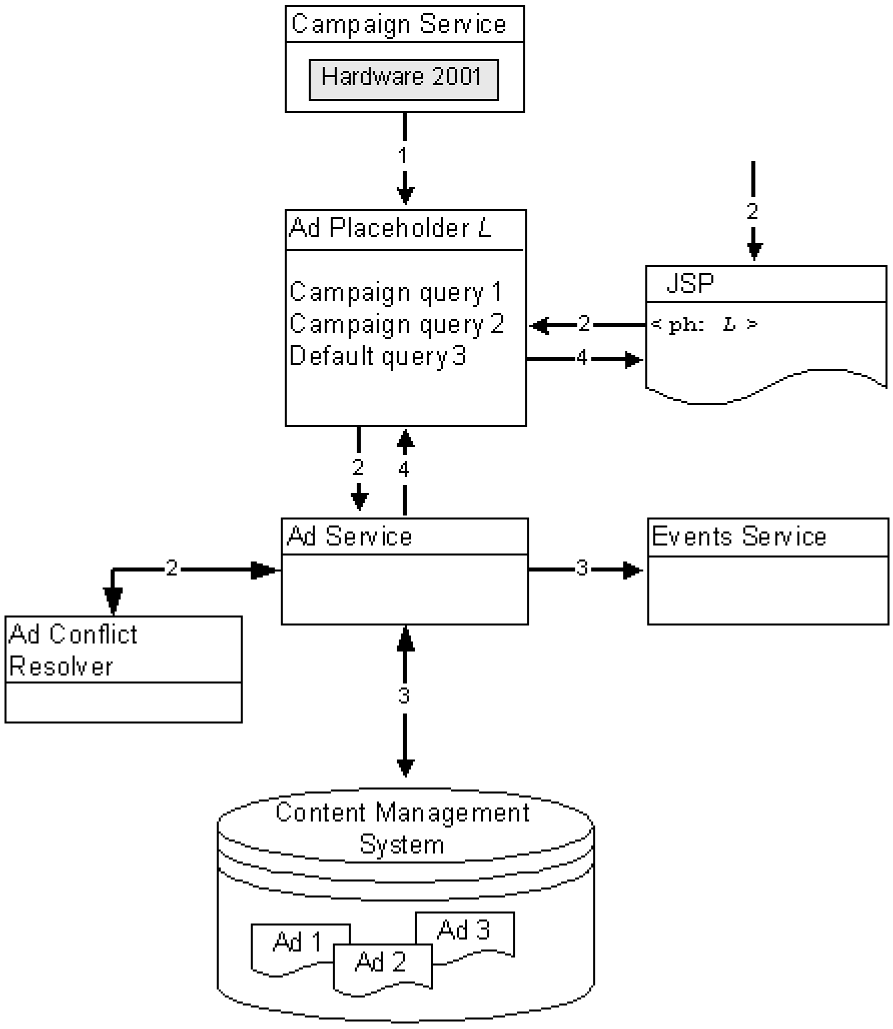
   For more information, refer to “Resolving Ad Query Conflicts” under “Working with Ad Placeholders” in the *Guide to Building Personalized Applications*.
3. The Ad Service does the following:
   a. It forwards the query to the content management system. If the query returns more than one ad, the ad placeholder uses the `adWeight` attribute of each ad to determine which one to retrieve.

   b. If the ad is associated with an active campaign, it determines whether the campaign has fulfilled its goal of displaying the ad a specific number of times. If the ad has already been displayed the specified number of times, the Ad Service selects another ad.

   c. It sends data to the Event Service indicating that the placeholder has displayed the ad. Notice that the Ad Service updates the ad-display count after it finds an ad but before it generates the HTML to display the ad. In some cases, for example, if a customer forces a browser to stop loading a page, the Ad Service increases the ad display count even though the ad was never displayed.

   For more information, refer to “How an Ad Placeholder Chooses from Ad Query Results” under “Working with Ad Placeholders” in the *Guide to Building Personalized Applications*.
4. The ad placeholder generates the HTML that the browser requires to display the ad content and places it in the JSP at the location of the placeholder tag.
5. If a customer clicks on the ad, the Ad Service redirects the URL and notifies the Event Service that a customer clicked the ad.

Figure 1-2 How Placeholders Display Ads for Campaigns



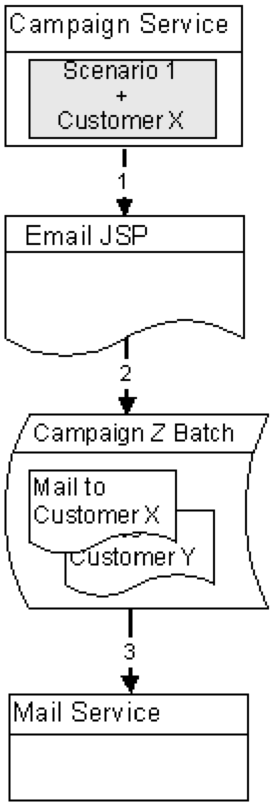
How Campaigns Use the Mail Service

Business Analysts (BAs) can specify that a scenario within a campaign sends an e-mail to a customer. For example, when a customer buys a flashlight, a scenario can send an e-mail that contains special offers for batteries.

When BAs create scenarios from the E-Business Control Center, they select a JSP file that contains the e-mail content. When a customer triggers the scenario action, the following processes occurs:

1. The Campaign Service uses internal HTTP to request the JSP that the scenario action specifies. In the request, it passes parameters to identify the name of the scenario and the identity of the customer who triggered the scenario action.
2. The JSP invokes any JSP tags that it contains, uses MIME types to encode non-ASCII output, and stores its output in the WebLogic Portal data repository. The data repository organizes the e-mails into batches; one batch for each campaign.
3. You issue a command to the Mail Service that specifies which batch of messages you want to send. The Mail Service uses the JavaMail API to send the messages.

Figure 1-3 The Mail Service for Campaigns



How Campaigns Offer Discounts

A BA creates discounts while defining actions for a scenario. When an event triggers a scenario to activate a discount the following process occurs:

1. The scenario action sends the following information to the Discount Association Service:

   - The customer's ID (customer PK), which is an identifier of the customer who the scenario determines is eligible for a discount
   - The discount's ID
   - The description for the discount, which a BA creates in the E-Business Control Center and which can be displayed in the shopping cart or other JSPs

   For example, a scenario states that between May 1 and May 10, all customers are eligible to purchase an ACME hand saw for $20. When the system clock reaches May 1 and Pat Gomes (a customer) logs in, the scenario engages and notifies the Discount Association Service that Pat Gomes is eligible for the discount.

   Another scenario states that all Gold Customers are eligible for a 10% discount on ACME drills. When Pat Gomes logs in, the Event Service sends a message to the Campaign Service. The Campaign Service determines that Pat Gomes is a Gold Customer. The scenario then engages and notifies the Discount Association Service that Pat Gomes is eligible for a 10% discount on ACME drills.

2. When a customer adds an item to the shopping cart, removes an item from the shopping cart, checks out, or confirms an order, the shopping cart places its data in a ShoppingCart Pipeline component.

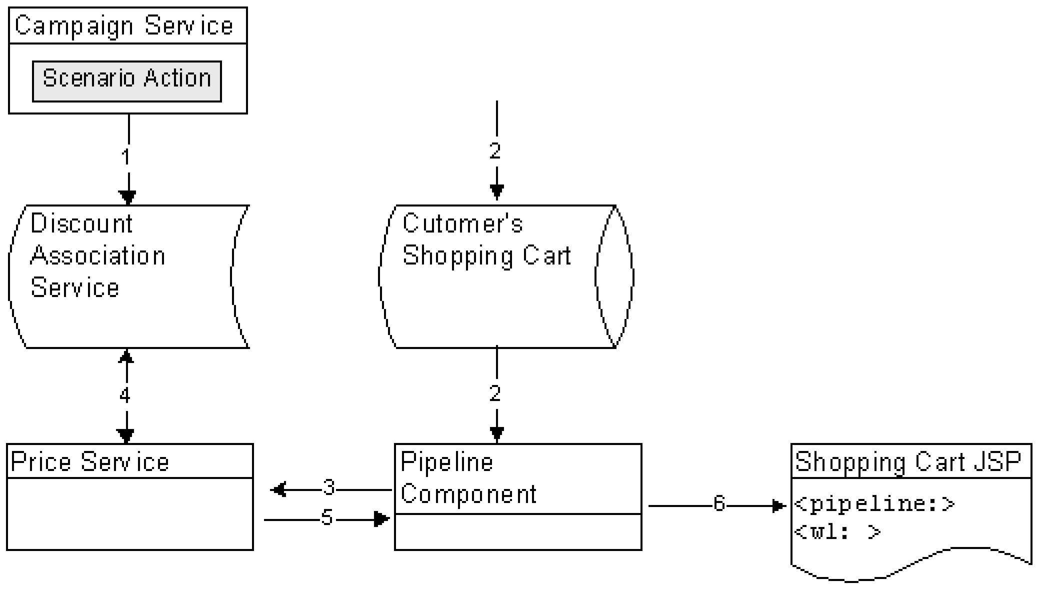
3. The ShoppingCart Pipeline component contacts the Pricing Service.

4. Using data from the Discount Association Service, the Shopping Cart Pipeline component, and from any third-party tax-calculation service, the Pricing Service calculates the total price of the customer's order. For more information on how the Pricing Service calculates the price of a customer's order, refer to the *Guide to Managing Purchases and Processing Orders*.

5. The Pricing Service sends its results to the PriceOrder Pipeline component.

6. Then the shopping cart JSP displays information from the Pipeline component.

Figure 1-4 How a Campaign Offers Discounts



Workflow for Developing Campaign Infrastructure

To develop campaign infrastructure, you set up data structures that the BA uses to define and run individual campaigns. To support a specific campaign, a BA might require you to add or update data structures and remove them when the campaign ends.

The following steps outline the process for developing campaign infrastructure:

1. **Define Custom Events**. WebLogic Portal includes a set of default events that trigger campaign actions. If a BA wants a campaign to respond or react to other events, you must define custom events. For more information, refer to “Supporting Custom Events for Campaigns” on page 3-40 and “Creating Custom Events” in the *Guide to Events and Behavior Tracking*.

2. **Set Up Ads and Ad Attributes**. You must load any advertising documents (ads) that a campaign displays into your content management system. To support ad placeholder queries, you must define attributes for each ad. To support ad clickthroughs and popup windows for clickthrough targets, and to set preferences for Shockwave movie files, you must attach an additional set of attributes to the image and Shockwave advertising documents.

   On any page that displays ads for a campaign, you must use a `<ph:placeholder>` JSP tag. Placeholder tags run queries that a BA constructs in the E-Business Control Center and display the query results.

3. **Add customer profile data to the session**. A customer profile is a key piece of information for determining whether a campaign scenario applies to a specific event. On any JSP that generates or reacts to events for a campaign, you must retrieve the customer's profile and place it in the `session` object by doing any of the following:

   - Use Webflow components.
   - If your Web application uses the `FORM` method for login, you can register the `P13NAuthFilter`, which initializes customer profiles when a customer logs in without using Webflow.
   - Use JSP tags, such as `<um:getProfile>`
   - Use APIs

   For more information, refer to "Initializing the Customer Profile" on page 3-38.

4. **Set Up the E-mail Service Properties and Campaign E-mail JSPs**. If the campaign sends e-mail, you must set properties for the e-mail service. Then you set up JSPs to contain the campaign's e-mail messages. You can use JSP tags and APIs to personalize the letter. Instead of using the WebLogic Portal e-mail service, you can set up an integration with a third-party e-mail service.

5. **Send Bulk E-mails**. The e-mail service places all e-mails in a batch. You must periodically use a command to send the batch. You can also use `cron` or any other scheduler that your operating system supports to issue the send-mail command.

6. **Clean Up**. After the campaign ends, you can remove any campaign-specific placeholders, JSPs, and JSP tags.

CHAPTER

2 Setting Up Ads for Campaigns

An ad is a document in your content management system that an ad placeholder displays. Ads can be an integral part to a campaign. For example, campaigns can specify as a goal to record a specific number of ad clickthroughs.

This topic includes the following sections:

- Describing the Ads in Your Content Management System
- Specifying Display and Clickthrough Behavior
- Loading Ads Into Your Content Management System
- Supporting Additional MIME Types

For more information about ads, refer to “Working with Ad Placeholders” in the *Guide to Building Personalized Applications*.

Describing the Ads in Your Content Management System

The queries that a BA defines for ad placeholders search through the descriptions (attributes) that you attach to the documents in your content management system. WebLogic Portal places no restrictions on the set of attributes that you use to describe your ads. For example, you can create attributes that describe the name of the product that the document advertises, the name of the ad sponsor, and a product category that matches the categories in your e-commerce product catalog.

We recommend that a BA analyses your advertising strategy and proposes a set of attributes that describe the ads in your content management system.

For information on adding attributes, refer to the documentation for your content management system. If you use the reference content management system, refer to “Loading Ads into the Reference Content Management System” on page 2-27.

Specifying Display and Clickthrough Behavior

Ad placeholders use a set of document attributes that you define in your content management system to support the following features:

- Choosing a single document if a query returns multiple documents
- Making an image ad clickable
- Supplying movie preferences for a Shockwave file

For information about associating attributes with documents, refer to the documentation for your content management system. If you use the reference content management system, refer to “Loading Ads into the Reference Content Management System” on page 2-27.

Table 2-1 describes the `adWeight` attribute, which you can associate with XHTML, image, and Shockwave documents.

Table 2-1 Attributes for All Document Types

Attribute Name	Value Type	Description and Recommendations
adWeight	Integer	Provides an integer that is used to select a document if a query returns multiple documents. Assign a high number to ads that you want to have a greater chance of being selected. For more information, refer to “How an Ad Placeholder Chooses from Ad Query Results” under “Working with Ad Placeholders” in the *Guide to Building Personalized Applications*. The default value for this attribute is `1`. **Note:** In the E-Business Control Center, you can assign a priority to a query for a scenario action. The priority, which bears no relation to the `adWeight` attribute, gives a greater or lesser chance that a placeholder runs a query. The `adWeight` attribute is used to choose an ad after a query has run. For more information, refer to “How the Ad Conflict Resolver Chooses a Query” under “Working with Ad Placeholders” in the *Guide to Building Personalized Applications*.

Table 2-2 describes attributes in addition to the `adWeight` attribute that you can associate with image files.

Table 2-2 Attributes for Image Files

Attribute Name	Value Type	Description and Recommendations
adTargetUrl	String	Makes an image clickable and provides a target for the clickthrough, expressed as a URL. The Event Service records the clickthrough. Use either `adTargetUrl`, `adTargetContent`, or `adMapName`, depending on how you want to identify the destination of the ad clickthrough.
adTargetContent	String	Makes an image clickable and provides a target for the clickthrough, expressed as the content management system's content ID. The Event Service records the clickthrough. Use either `adTargetUrl`, `adTargetContent`, or `adMapName`, depending on how you want to identify the destination of the ad clickthrough.
adMapName	String	Makes an image clickable, using an image map to specify one or more targets. The value for this attribute is used in two locations: ■ In the anchor tag that makes the image clickable, `<a href=`*`value`*`> <img> </a>` ■ In the map definition, `<map name=`*`value`*`>` Use either `adTargetUrl`, `adTargetContent`, or `adMapName`, depending on how you want to identify the destination of the ad clickthrough. If you specify a value for `adMapName`, you must also specify a value for `adMap`.
adMap	String	Supplies the XHTML definition of an image map. If you specify a value for `adMap`, you must also specify a value for `adMapName`.
adWinTarget	String	Displays the target in a new pop-up window, using JavaScript to define the pop-up. The only value supported for this attribute is `newwindow`.

Table 2-2 Attributes for Image Files (Continued)

Attribute Name	Value Type	Description and Recommendations
adWinClose	String	Specifies the name of a link that closes a pop-up window. The link appears at the end of the window content. For example, if you provide “Close this window” as the value for this attribute, then “Close this window” appears as a hyperlink in the last line of the pop-up window. If a customer clicks the link, the window closes.
adAltText	String	Specifies a text string for the `alt` attribute of the `<img>` tag. If you do not include this attribute, the `<img>` tag does not specify an `alt` attribute.
adBorder	Integer	Specifies the value for the `border` attribute of the `<img>` tag. If you do not include this attribute, the `border` attribute is given a value of `"0"`.

Table 2-3 describes attributes in addition to the `adWeight` attribute that you can associate with Shockwave files. Ad placeholders and the `<ad:adTarget>` tag format these values as attributes of the `<OBJECT>` tag, which Internet Explorer on Windows uses to display the file, and the `<EMBED>` tag, which browsers that support the Netscape-compatible plug-in use to display the file.

For more information about these attributes, refer to your Shockwave developer documentation.

Table 2-3 Attributes for Shockwave Files

Attribute Name	Value Type	Description and Recommendations
swfLoop	String	Specifies whether the movie repeats indefinitely (`true`) or stops when it reaches the last frame (`false`). Valid values are `true` or `false`. If you do not define this attribute, the default value is `true`.
swfQuality	String	Determines the quality of visual image. Lower qualities can result in faster playback times, depending on the client’s Internet connection. Valid values are `low`, `high`, `autolow`, `autohigh`, `best`.

Table 2-3 Attributes for Shockwave Files (Continued)

Attribute Name	Value Type	Description and Recommendations
swfPlay	String	Specifies whether the movie begins playing immediately on loading in the browser. Valid values are `true` or `false`. If you do not define this attribute, the default value is `true`.
swfBGColor	String	Specifies the background color of the movie. This attribute does not affect the background color of the HTML page. Valid value syntax is #*RRGGBB*.
swfScale	String	Determines the dimensions of the movie in relation to the area that the HTML page defines for the movie. Valid values are `showall`, `noborder`, `exact fit`.
swfAlign	String	Determines whether the movie aligns with the center, left, top, right, or bottom of the browser window. If you do not specify a value, the movie is aligned in the center of the browser. Valid values are `l`, `t`, r, `b`.
swfSAlign	String	Determines the movie’s alignment in relation to the browser window. Valid values are `l`, `t`, `r`, `b`, `tl`, `tr`, `bl`, `br`.
swfBase	String	Specifies the directory or URL used to resolve relative pathnames in the movie. Valid values are `.(period)`, *`directory-name`*, *URL*.
swfMenu	String	Determines whether the movie player displays the full menu. Valid values are `true` or `false`.

Loading Ads Into Your Content Management System

This section contains the following subsections:

- Loading Ads into a Third-Party Content Management System
- Loading Ads into the Reference Content Management System

Loading Ads into a Third-Party Content Management System

You use the same procedure for loading ads into your content management system as you use for loading any other document. For information on loading documents, refer to the documentation for your content management system.

For more information about using a content management system with WebLogic Portal, refer to “Creating and Managing Content” in the *Guide to Building Personalized Applications*.

Loading Ads into the Reference Content Management System

WebLogic Portal provides a content management system for sites with limited content-management needs. If you use the reference content management system, you must load ads and ad attributes at the same time. You cannot add attributes to documents that have already been loaded.

When you install WebLogic Portal, the reference content management system (which uses the sample Cloudscape database) already contains a set of sample ads. If you set up other supported databases, refer to “Deploying Your Business Data” in the *Deployment Guide*, which describes using the `loadSampleData` script to populate your database and the reference content management system with sample data. This section describes loading ads into the reference management system in addition to any ads that you loaded into the system using the `loadSampleData` script.

Note: The reference content management system requires different processes for loading ads and loading other documents that you do not use as advertisements. This section describes only the process of loading ads. For information on loading other documents, refer to “Creating and Managing Content” in the *Guide to Building Personalized Applications*.

To load ads and ad attributes into the reference content management system, you must do the following:

- Set Up Attributes in HTML Documents
- Set Up Attribute Files for Image and Shockwave Documents
- Move Files Into the dmsBase/Ads Directory Tree
- Run the loadads Script

Set Up Attributes in HTML Documents

For ads that contain only HTML, you must place document attributes in `<META>` tags within a document's `<HEAD>` element. Use the following syntax in the `<META>` tag:

```
<META name="attribute-name" content="attribute-value">
```

Use a separate `<META>` tag for each document attribute. For example:

```
<META name="attribute1-name" content="attribute1-value">
<META name="attribute2-name" content="attribute2-value">
<META name="attribute3-name" content="attribute3-value">
```

Listing 2-1 shows an HTML file that contains a simple ad with several attributes.

Listing 2-1   Attributes for an HTML Ad

```
<HTML>
<HEAD>

<META name="adWeight" content="3">
<META name="productCategory" content="hardware">
<META name="productSubCategory" content="electic drill">
<META name="productName" content="Super Drill">
<META name="Manufacturer" content="ACME">

</HEAD>

<BODY>

<P>Buy our Super Drill. It’ll get the job done!</P>

</BODY>

</HTML>
```

Set Up Attribute Files for Image and Shockwave Documents

For ads that are images or Shockwave movies, you must place attributes in a separate file. Each image or Shockwave file must be accompanied by a separate file that is named with the following convention:

`filename.extension``.md.properties`

Both files must be located in the same directory.

For example, for an image file named `superDrill.jpg`, you must place attributes in a file named `superDrill.jpg.md.properties`.

Within the *`filename.extension`*`.md.properties` file, use the following syntax to express attributes and values:

`attribute-name``=`*`attribute-value`*

Listing 2-2 shows an example file that contains attributes for an image ad.

Listing 2-2 Syntax for the Attributes File

```
adWeight=5
adTargetUrl=AcmeAds/saws.jpg
adAltText=Buy ACME and save!


productCategory=hardware
productSubCategory=electic drill
productName=Super Drill
Manufacturer=ACME
```

Move Files Into the dmsBase/Ads Directory Tree

All HTML, image, and Shockwave files, and all attributes files must be located below the following directory:

`PORTAL_HOME/dmsBase/Ads`

where `PORTAL_HOME` is the directory in which you installed WebLogic Portal.

You can place documents in subdirectories of the `Ads` directory, though the reference content management system does not use the subdirectories to organize documents.

If you use subdirectories to manage your source files, you must place the attributes files in the same directory as the files that they describe. For example, `superDrill.jpg` and `superDrill.jpg.md.properties` must be in the same directory.

Run the loadads Script

The `loadads` script loads documents from the `dmsBase/Ads` directory to the content management system. It also attaches attributes to the documents.

The pathname for the script is as follows:

`PORTAL_HOME\bin\win32\loadads.bat` (Windows)
`PORTAL_HOME/bin/unix/loadads.sh` (UNIX)

where `PORTAL_HOME` is the directory in which you installed WebLogic Portal.

For more information on loading documents into the reference content management system, refer to “Creating and Managing Content” in the *Guide to Building Personalized Applications*.

Supporting Additional MIME Types

To display an ad, placeholders refer to a document's MIME type and then generate the HTML tags that a browser requires for the specific document type. For example, to display an image-type document, an ad placeholder must generate the `<img>` tag that a browser requires for images. By default, ad placeholders can generate the appropriate HTML only for the following MIME types:

- XHTML (a fragment or an entire document). For this type of document, a placeholder passes the text directly to the JSP.
- Images. For this type of document, a placeholder generates an `<img>` tag with attributes that the browser needs to display the image. If you want images to be clickable, you must specify the target URL and other link-related information as ad attributes in your content management system.
- Shockwave files. For this type of document, a placeholder generates the `<OBJECT>` tag, which Microsoft Internet Explorer on Windows uses to display the file, and the `<EMBED>` tag, which browsers that support the Netscape-compatible plug-in use to display the file. In your content management system, you can specify attributes for the `<OBJECT>` and `<EMBED>` tags.

If you are familiar with basic Java programming, you can write classes that enable placeholders to generate HTML for additional MIME types. To support additional MIME types, you must complete the following tasks:

- Create and Compile a Java Class to Generate HTML
- Register the New Class

Create and Compile a Java Class to Generate HTML

To generate the HTML that the browser requires to display the MIME type, create and compile a Java class that implements the `com.bea.p13n.ad.AdContentProvider` interface. For information on this interface, refer to *WebLogic Portal Javadoc*.

After you compile the class, you must make sure the class is available to the application. One way to do this is to add the class appropriately to one of the deployed jar files, such as `placeholder.jar` or your own jar file. Another way to make the class available to the application is to save it under a directory that is specified in the system's `CLASSPATH` environment variable. For example, create a `WL_PORTAL_HOME/classes` directory and add it to the set-environment script. For more information about the `CLASSPATH` environment variable, refer to "Setting Environment Variables," under "Starting and Shutting Down the Server" in the *Deployment Guide*.

Register the New Class

After you save the class in a directory that is in your classpath, you must notify WebLogic Portal of its existence:

1. Stop the WebLogic Portal instance that is running your application. For information on stopping a server, refer to "Starting and Shutting Down a Server" in the *Deployment Guide*.
2. Create a backup copy of `PORTAL_HOME/application/`*`your-application`*`/META-INF/application-config.xml`
3. Open `application-config.xml` in a text editor and find the `<AdService>` element.
4. Add the following as a subelement of `<AdService>`:

   ```
   <AdContentProvider
           Name="MIME-type"
           Provider="name-of-your-class"
           Properties="optional-properties-for-your-class"
         >
   </AdContentProvider>
   ```

   Provide the following values for the attributes of the `AdContentProvider` element:

   - `Name`. The name of the MIME type that you want to support.
   - `Provider`. The name of the compiled Java file. If you saved the file below a directory that your `CLASSPATH` environment variable names, you must include the file's pathname, starting one directory level below the directory in classpath.

- `Properties`. Any additional properties or parameters want to pass to your object.

For example, if you added `WL_PORTAL_HOME/classes` to the system classpath, save your class to support AVI files as `WL_PORTAL_HOME/classes/myclasses/MimeAvi.class`.

Alternately, if you have already deployed a JAR of your own Java classes called `myServices.jar`, add `myclasses/MimeAvi.class` to that JAR file.

To register your classname, add a subelement to the `AdService` element as illustrated in Listing 2-3.

Listing 2-3 Add An AdContentProvider Element

```
<AdService
      Name="wlcsApp"
      DisplayFlushSize="10"
      Rendering="com.bea.p13n.ad.AdContentProviderBase"
      EventTracker="com.bea.campaign.AdTracking"
      AdClickThruURI="AdClickThru"
      ShowDocURI="ShowDoc"
    >

      <AdContentProvider
        Name="text"
        Provider="com.bea.p13n.ad.render.TextContentProvider"
        Properties=""
      >
      </AdContentProvider>

      <AdContentProvider
        Name="image"
        Provider="com.bea.p13n.ad.render.ImageContentProvider"
        Properties="AdClickThruURI=AdClickThru;ShowDocURI=ShowDoc"
      >
      </AdContentProvider>

      <AdContentProvider
        Name="application/x-shockwave-flash"
        Provider="com.bea.p13n.ad.render.ShockwaveContentProvider"
        Properties="ShowDocURI=ShowDoc"
      >
      </AdContentProvider>

      <AdContentProvider
        Name="video/x-msvideo"
        Provider="myclasses.MimeAvi"
        Properties=""
      >
      </AdContentProvider>

  </AdService>
```

5. Save your modifications to `application-config.xml`.
6. Restart WebLogic Portal.

CHAPTER

3 Setting Up JSP Tags and Scriptlets for Campaigns

Campaigns require a minimal set of JSP tags and scriptlets to support their services. In some cases, you might have already added the required JSP tags to implement other WebLogic Portal services.

This topic includes the following sections:

- Initializing the Customer Profile
- Supporting Custom Events for Campaigns
- Configuring Support for Ad Placeholders
- Using Ad Placeholder Tags to Display Ads
- Using the <ad:adTarget> JSP Tag to Display Ads
- Creating Scriptlets to Display Discounts

For information about the JSP tags that support the e-mail service, refer to Chapter 4, “Setting Up and Sending E-mail for Campaigns.”

Initializing the Customer Profile

A customer profile is a key piece of information for determining whether a campaign scenario applies to a specific event. On any JSP that generates or reacts to events for a campaign, you must initialize a customer's profile by retrieving it from the RDBMS or Unified User Profile and placing it in the `HttpSession` object.

You can do any of the following to initialize a customer profile:

- Using Webflow Components
- Using P13NAuthFilter
- Initializing Profiles with JSP Tags or APIs

Using Webflow Components

You can use Webflow components to initialize customer profiles.

If you are using the sample `portal` application as a starting point for your development, the `PostLoginProcessor` Webflow component initializes customer profiles as part of the customer login process. The `PostLoginProcessor` is in the `stockportal` Web application and is invoked by the `security` namespace. You can use the E-Business Control Center to view the properties of `PostLoginProcessor`. For more information, refer to "Customizing Portlets and Portals" in the *Getting Started with Portals and Portlets*.

You can create your own Webflow processor component to initialize customer profiles, and you can place this component at any location in a Web application's Webflow. Because many other JSP tags and WebLogic Portal services rely on data from the customer profile, consider making your processor component part of the customer login process. To successfully initialize the customer profile, the processor must retrieve the profile and place it in the `HTTPSession` object.

Using P13NAuthFilter

If your Web application uses the `FORM` method for login, you can register the `P13NAuthFilter`, which initializes customer profiles without using Webflow. `P13NAuthFilter` is activated when a customer logs in.

To specify the `FORM` login method, add the following elements to your Web application's `web.xml` file:

```
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
   <form-login-page>login-JSP</form-login-page>
   <form-error-page>login-error-page</form-error-page>
  </form-login-config>
</login-config>
```

To register the `P13NAuthFilter`, add the following element to your Web application's Web application's `weblogic.xml` file:

```
<auth-filter>com.bea.p13n.servlets.P13NAuthFilter</auth-filter>
```

For more information, refer to the following:

- For information about editing your Web application's `web.xml` and `weblogic.xml` files, refer to "Assembling Your Web Application" in the *Deployment Guide*.
- For information about the FORM login method, refer to "Configuring Security in Web Applications" in the *WebLogic Server Assembling and Configuring Web Applications* guide.
- For information about `com.bea.p13n.servlets.P13NAuthFilter`, refer to the *WebLogic Portal Javadoc*.

Initializing Profiles with JSP Tags or APIs

Instead of the preceding approaches, you can initialize profiles by adding the `<um:getProfile>` JSP tag to a JSP. For more information, refer to `<um:getProfile>` under "JSP Tag Library Reference" in the *Guide to Building Personalized Applications*.

You can also use `com.bea.p13n.usermgmt.profile.ProfileFactory` to retrieve the user profile and `com.bea.p13n.usermgmt.SessionHelper` to put it in the `HttpSession`. For more information, refer to the *WebLogic Portal Javadoc*.

Supporting Custom Events for Campaigns

The Event service provides a set of JSP tags that you use to specify the user behavior you are interested in monitoring. As users navigate across your site, these tags generate events, which can trigger actions in a campaign.

You can create custom events and JSP tags to extend the set of events to which a campaign can respond. Note following requirements for your custom JSP tag:

- It must include a `user-id` attribute. The value of this attribute must identify the user who triggered the event.
- If you include a `request` attribute, its value must be set to a valid `com.bea.p13n.http.Request` object (which is a copy of `HttpServletRequest`). If you do not include this `request` attribute, then customer segments or scenario actions that evaluate data in the `Request` or `Session` property sets will never be triggered; additionally, the customer will not be associated with any group profile and any default values that come from the group profile will not be set.

To set the `user-id` and `request` attributes for your custom JSP tags, add the following scriptlets to the JSP:

```
<%@ page import="com.bea.p13n.events.Event"%>
<%@ page import="com.bea.p13n.tracking.TrackingEventHelper"%>
<%@ page import="com.bea.p13n.usermgmt.SessionHelper"%>

<%
   Event evt = new Event("MyCustomEvent");
   evt.setAttribute("user-id",
   SessionHelper.getUserId(request));
   evt.setAttribute("request", new Request(request, true));
   TrackingEventHelper.dispatchEvent(request, evt);
 %>
```

For more information about using events with campaigns, refer to the *Guide to Events and Behavior Tracking*.

Configuring Support for Ad Placeholders

For the `<ph:placeholder>` and `<ad:adTarget>` tags to work correctly, a Web application requires the following configuration:

- EJB references to `ejb/PlaceholderService`, `ejb/AdBucketService`, and `ejb/DocumentManager` must be specified in the Web Application’s `web.xml` and `weblogic.xml` files.
- The `com.bea.p13n.ad.servlets.AdClickThruServlet` must be mapped to `/AdClickThru/*` in the Web Application’s `web.xml` file.
- The `com.bea.p13n.content.servlets.ShowDocServlet` must be mapped to `/ShowDoc/*` in the Web Application’s `web.xml` file.

For more information, refer to “Assembling Your Web Application” in the *Deployment Guide*. Also, refer to the `web.xml` and `weblogic.xml` files in `WL_PORTAL_HOME/applications/portal/stockportal/WEB-INF`.

Using Ad Placeholder Tags to Display Ads

After a BA uses the E-Business Control Center to create ad placeholders, a BE creates ad placeholder tags in the Web site's JSPs. The placeholder definition determines the behavior of the placeholder tag.

You can create placeholders in JSPs that directly display content to a customer (for example, `index.jsp`) or in JSPs that are included in other JSPs (for example, `heading.jsp`).

For more information about ad placeholders, refer to "Working with Ad Placeholders" in the *Guide to Building Personalized Applications*.

To Create an Ad Placeholder Tag

1. In a text editor, open a JSP.
2. Import the tag library by adding the following tag near the top of the JSP:

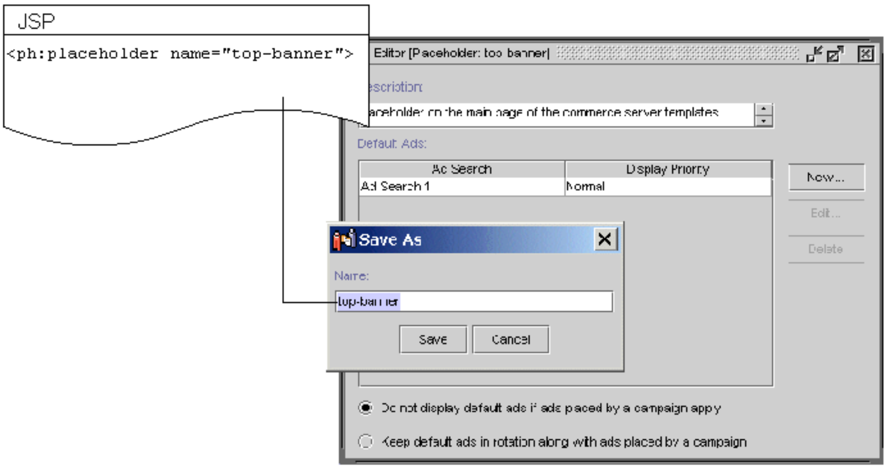
   ```
   <%@ taglib uri="ph.tld" prefix="ph" %>
   ```

3. Find the location in which the Business Analyst wants to display the ad.
4. Use the following syntax to use the placeholder tag:

   ```
   <ph: placeholder= "{ placeholder-name | scriptlet }" >
   ```

   where *`placeholder-name`* refers to the name of an existing placeholder definition (see Figure 3-1) or where *`scriptlet`* returns the name of an existing placeholder.

Figure 3-1 Placeholder Names Must Match


JSP
<ph:placeholder name="top-banner">
Editor [Placeholder: top banner]
escription:
aceholder on the main page of the commerce server templates
Default Ads:
Ad Search
Display Priority
Ad Search 1
Normal
New...
Edit...
Delete
Save As
Name:
top-banner
Save
Cancel
Do not display default ads if ads placed by a campaign apply
Keep default ads in rotation along with ads placed by a campaign

Listing 3-1 shows an example from the heading include file of the e-commerce sample JSP templates (`PORTAL_HOME\applications\wlcsApp\wlcs\commerce\includes\header.inc`).

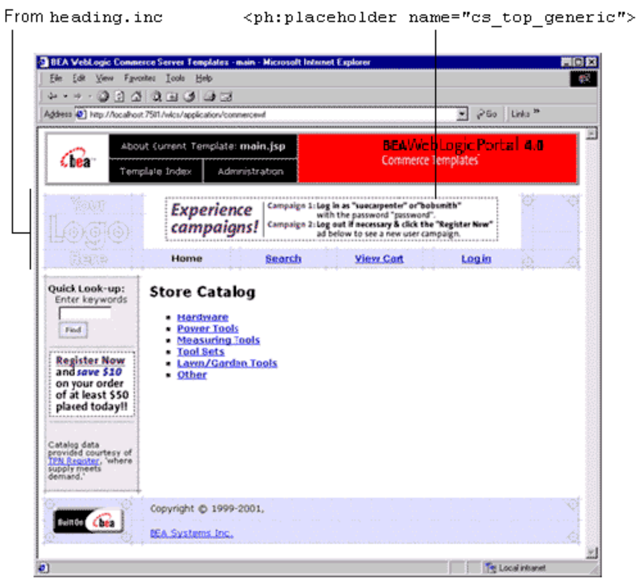
All JSP files in the sample wlcs Web application include `header.inc` to create consistency in the top banner. Instead of requiring the banner on each page to use the same placeholder, the placeholder in `heading.inc` uses a scriptlet to determine the value of the `name` attribute. A JSP can use the default value for the `name` attribute (which is `cs_top_generic`), or it can specify define a variable named `banner` and specify a placeholder name as the value for the variable.

Listing 3-1 Using a Scriptlet for the Placeholder Name

```
<%

   String banner = (String)pageContext.getAttribute("bannerPh");
   banner = (banner == null) ? "cs_top_generic" : banner;

%>

<!-- ------------------------------------------------------------ -->

<table width="100%" border="0" cellspacing="0" cellpadding="0" height="108">

  <tr><td rowspan="2" width="147" height="108">
  <img src="<%=WebflowJSPHelper.createGIFURL(request, response,
  "/commerce/images/header_logo.gif")%>" width="147" height="108"></td>

  <td colspan="7" height="75" align="center" valign="middle">


<ph:placeholder name="<%= banner %>" />

</td>
```

Figure 3-2 illustrates how WebLogic Portal renders the placeholder in the `main.jsp` file, which is the home page for the sample wlcs Web application.

Figure 3-2 Placeholder in the E-Commerce JSP Templates



For more information about the `<ph:placeholder>` tag, refer to "JSP Tag Library Reference" in the *Guide to Building Personalized Applications.*

Using the <ad:adTarget> JSP Tag to Display Ads

The `<ad:adTarget>` JSP tag is an additional mechanism for selecting and displaying ads. Use `<ad:adTarget>` if it is essential that a specific query run in a specific location.

Like an ad placeholder, `<ad:adTarget>` can do the following:

- Generate the HTML that a browser requires to display the types of documents that are described in "Supporting Additional MIME Types" on page 2-32.

- Use the document attributes that are described in “Specifying Display and Clickthrough Behavior” on page 2-22.
- Use the Ad Service to choose an ad if a query returns multiple documents, as described in “How an Ad Placeholder Chooses from Ad Query Results” under “Working with Ad Placeholders” in the *Guide to Building Personalized Applications*.

However, the `<ad:adTarget>` is **unlike** ad placeholders in the following ways:

- It contains its own query; it does not refer to a definition that a BA creates in the E-Business Control Center. If you want to change the query, you modify the tag in the JSP.
- A campaign scenario cannot specify a query to run in an `<ad:adTarget>` tag. Scenarios can only use ad placeholders to run queries.
- Because it contains only a single query, it does not need to use the Ad Conflict Resolver as described in “How the Ad Conflict Resolver Chooses a Query” under “Working with Ad Placeholders” in the *Guide to Building Personalized Applications*.

For a more information about `<ad:adTarget>`, refer to “JSP Tag Library Reference” in the *Guide to Building Personalized Applications*.

Creating Scriptlets to Display Discounts

The Pricing Service calculates the effect of discounts on a customer’s order. To display the discount information that the Pricing Service calculates, you can use the following objects and their methods:

- `DiscountPresentation`
- `OrderAdjustment` and its associated `AdjustmentDetail`

This section provides the following subsections:

- The Sequence of Applying Discounts in the Shopping Cart
- The DiscountPresentation Object

- The OrderAdjustment and AdjustmentDetail Objects
- Example of a Discount for the Order
- Example of a Discount for the Shipping Charges

The Sequence of Applying Discounts in the Shopping Cart

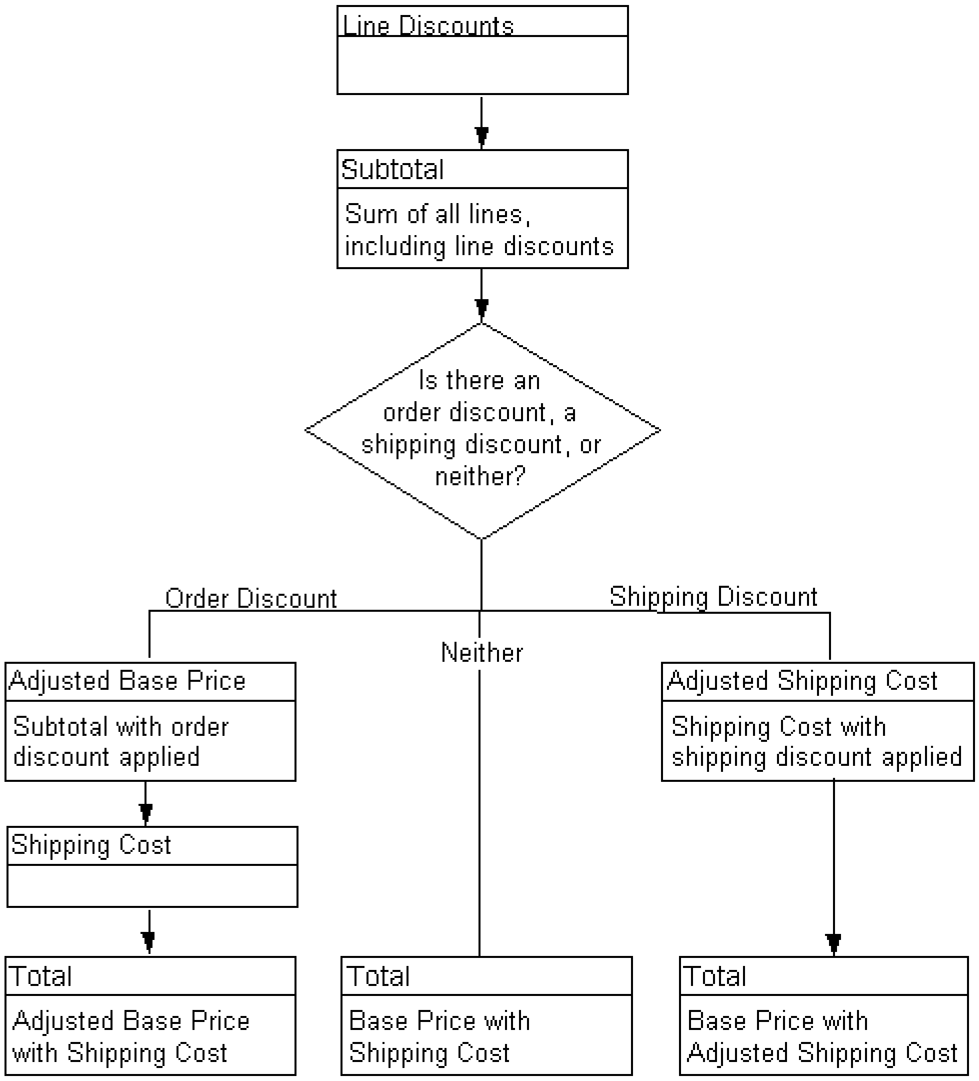
The Pricing Service follows this sequence for applying discounts (see Figure 3-3):

1. It applies item discounts to each line in the shopping cart.
2. It calculates a subtotal by adding the price of all lines in the cart.
3. It calculates an adjusted base price by applying any order discounts to the subtotal (if any order discounts apply).
4. It calculates a total by applying the shipping cost (including any shipping discounts) to the adjusted base price.

The following formula summarizes the discount-application process:

(Sum of line items) + order discounts (if any) + shipping (including any shipping discounts) = total

Figure 3-3 Discount Sequence


Line Discounts
Subtotal
Sum of all lines, including line discounts
Is there an order discount, a shipping discount, or neither?
Order Discount
Shipping Discount
Neither
Adjusted Base Price
Subtotal with order discount applied
Shipping Cost
Adjusted Shipping Cost
Shipping Cost with shipping discount applied
Total
Adjusted Base Price with Shipping Cost
Total
Base Price with Shipping Cost
Total
Base Price with Adjusted Shipping Cost

The DiscountPresentation Object

The shopping cart uses a *line* to represent the quantity of a single catalog item that a customer places in the cart. For example, if a customer orders 1 item, the shopping cart contains 1 line. If the customer changes the quantity of the item to 2, the shopping cart still contains 1 line.

For each shopping cart line that receives a discount, the Pricing Service instantiates a `DiscountPresentation` object to describe the discount. If multiple discounts apply to the line, then the Pricing Service instantiates multiple `DiscountPresentation` objects. (A line may receive 1 or more discounts, up to the quantity of the line.)

This section includes the following subsections:

- Statement to Import the Java Class
- shoppingCartLine Method for Retrieving the Object
- The DiscountPresentation Methods
- Example of Discount for Items in the ShoppingCartLine

Statement to Import the Java Class

To display the information in the `DiscountPresentation` object on a given JSP, you import the class by including the following statement:

```
<%@ page
import="com.bea.commerce.ebusiness.price.service.DiscountPresenta
tion" %>
```

shoppingCartLine Method for Retrieving the Object

From any JSP on which you want to display information about discounts for a shopping cart line, you use the following `shoppingCartLine` method:

- `getDiscountPresentations()`

  The method returns a `java.util.List` object, which contains a list of `DiscountPresentation` objects.

The DiscountPresentation Methods

After you use `getDiscountPresentations()`to retrieve a list of `DiscountPresentation` objects, you can call `DiscountPresentation` methods to retrieve discount information. Table 3-1describes the methods.

Table 3-1 DiscountPresentation Methods

Method	Returns Data of Type	Description
`getQuantity()`	Double	The number of items in the line that are discounted.
`getUnitPrice()`	Money	The discounted price of a single item. The `com.beasys.axiom.units.Money` class defines the Money data type.
`getDiscount()`	Money	The total price reduction from this discount: (Number of discounted items) X (The discounted unit price.)
`getComputation()`	String	The method of computation for the discount. For example, `50% off`.
`getReason()`	String	The reason for the discount as described in the E-Business Control Center. For example, `Spring Sale`.
`getShortText()`	String	A concatenation of `getComputation()` and `getReason()`. For example, the method can return `Discount: 50% off - Spring Sale`.

For more information on `com.bea.commerce.ebusiness.price.service.DiscountPresentation`, refer to the *WebLogic Portal Javadoc*.

Example of Discount for Items in the ShoppingCartLine

For purposes of this example, assume that a customer’s shopping cart contains only one line with the following information:

```
quantity 5 hammer-71-UF30588 MSRP: $18.00, "Our Price": $10.00
```

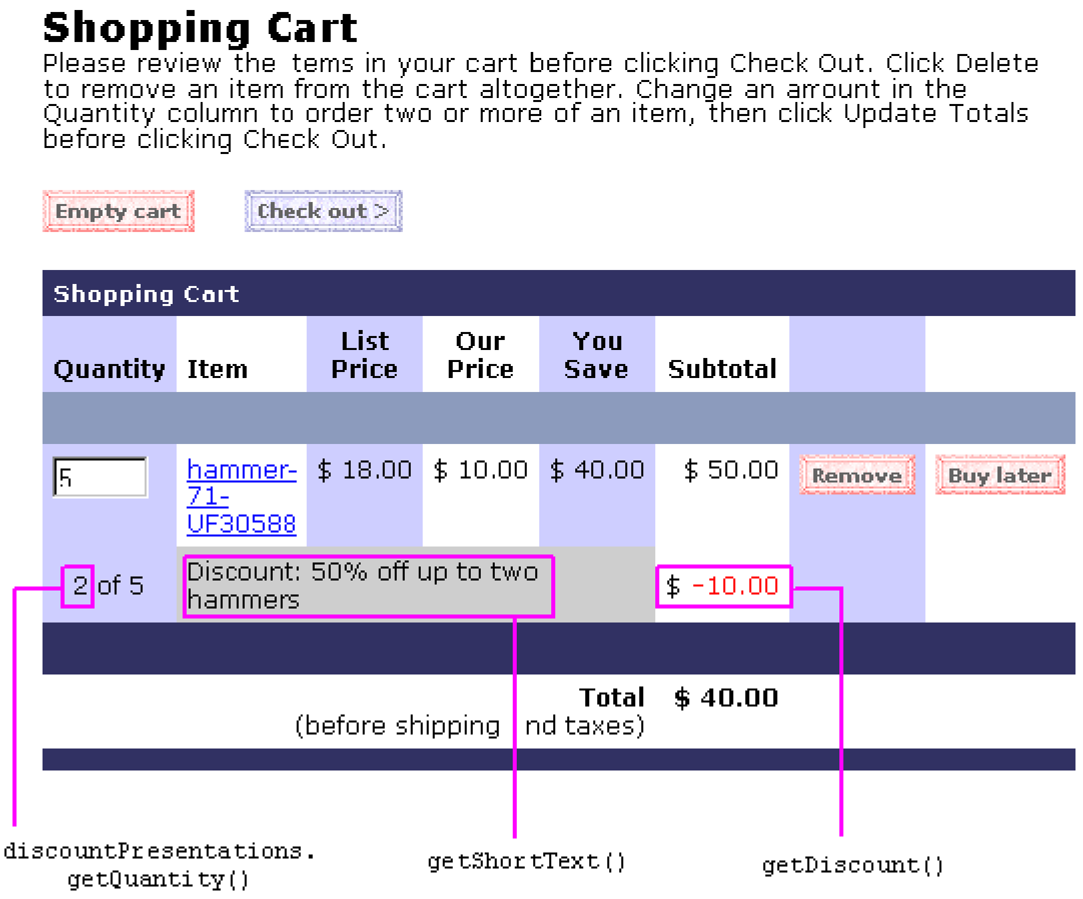
The Pricing Service determines that the line is eligible for one discount that offers 50% off the price of up to two hammers. It applies the discount and creates a `DiscountPresentation` object that contains the following information:

- `getQuantity()`: 2. Only two hammers are eligible for the discount
- `getUnitPrice()`: $5.00. Each hammer is $5.00 after the discount
- `getDiscount()`: $10.00. After applying the discount to two hammers, the total discount amount is $10.00 (2 (.5 x $10.00))
- `getComputation()`:"50% off"
- `getReason()`: "up to two hammers"
- `getShortText()`: "Discount: 50% off up to two hammers"

The shopping-cart subtotal is: $40.00 (3 x $10.00 + 2 x $5.00)

Figure 3-4 illustrates a shopping cart that displays the line discount.

Figure 3-4 Discount on Shopping Cart Line



Shopping Cart

Please review the tems in your cart before clicking Check Out. Click Delete to remove an item from the cart altogether. Change an arrount in the Quantity column to order two or more of an item, then click Update Totals before clicking Check Out.

Quantity	Item	List Price	Our Price	You Save	Subtotal		
5	hammer-71-UF30588	$ 18.00	$ 10.00	$ 40.00	$ 50.00	Remove	Buy later
2 of 5	Discount: 50% off up to two hammers				$ -10.00		
				Total (before shipping nd taxes)	$ 40.00		

The OrderAdjustment and AdjustmentDetail Objects

The Pricing Service always instantiates at least one `OrderAdjustment` object to describe the shipping cost that is associated with an order. It also instantiates at least one `AdjustmentDetail` object to provide additional information about the base shipping cost. If an order includes a discount on shipping cost, the Pricing Service creates a second `AdjustmentDetail` object to describe the shipping discount.

If an order includes a discount that applies to the entire order (for example, 10% off the order subtotal), then the Pricing Service instantiates an `OrderAdjustment` object to describe the order discount. It also instantiates a single `AdjustmentDetail` object to provide additional information about the discount. (See Figure 3-5.)

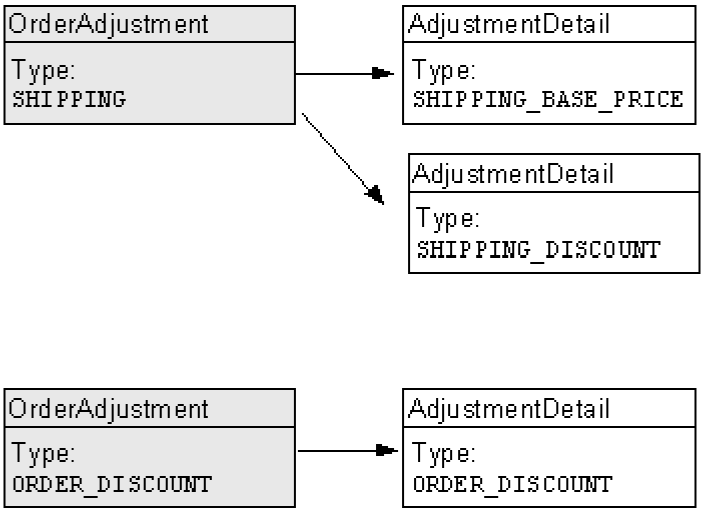
Note that a shopping cart may have an order discount or a shipping discount, but not both. For any given shopping cart with order or shipping discounts, the Price Service creates **one** of the following:

- An order discount `AdjustmentDetail` object that is attached to the order-discount `OrderAdjustment` object.

  or

- A shipping discount `AdjustmentDetail` object that is attached to the shipping `OrderAdjustment` object.

Figure 3-5 OrderAdjustment and AdjustmentDetails Objects



This section contains the following subsections:

- Statements to Import the Java Classes
- ShoppingCart Methods for Retrieving the Objects
- The OrderAdjustment Methods
- The AdjustmentDetail Methods
- Example of a Discount for the Order
- Example of a Discount for the Shipping Charges

Statements to Import the Java Classes

To display the information in the `OrderAdjustment` and `AdjustmentDetail` objects on a given JSP, you import the classes by including the following statements:

```
<%@ page
import="com.bea.commerce.ebusiness.price.quote.OrderAdjustment"
%>
<%@ page
import="com.bea.commerce.ebusiness.price.quote.AdjustmentDetail"
%>
```

ShoppingCart Methods for Retrieving the Objects

From any JSP on which you want to display information about discounts on shipping cost and discounts on the overall order, you use the following `ShoppingCart` methods to retrieve the `OrderAdjustment` and `AdjustmentDetails` objects:

Note: Before you can call these methods, you must first use the `PriceOrderPC` Pipeline component to set them.

- `getOrderDiscountPresentations()`. Use this method to retrieve a list of `OrderAdjustment` objects that describe discounts on the order subtotal. If there are no order discounts, this method returns a null value.
- `getShippingDiscountPresentations()`. Use this method to retrieve a list of `OrderAdjustment` objects that describe shipping costs and discounts.

Note that attached to each `OrderAdjustment` object is at least one `AdjustmentDetails` object, as described in “The OrderAdjustment and AdjustmentDetail Objects” on page 3-52.

The OrderAdjustment Methods

After you use the ShoppingCart methods to retrieve an `OrderAdjustment` object, you can use `OrderAdjustment` methods to retrieve discount or shipping information for the current JSP. Table 3-2 describes the methods that are of general interest to BEs.

Table 3-2 OrderAdjustment Methods

Method	Returns	Description
`getType()`	`AdjustmentType` object	One of the following types of adjustments: ■ `ORDER_DISCOUNT` ■ `SHIPPING` For an example of using `getType()` to determine the type of order adjustment, refer to Listing 3-2.
`getBasePrice()`	Money	The initial order price prior to applying the adjustment. The `com.beasys.axiom.units.Money` class defines the Money data type.
`getActualPrice()`	Money	The shipping price or order subtotal after applying discounts.
`getAdjustmentAmount()`	Money	The order price after applying the adjustment.
`getDetails()`	List	The list of `AdjustmentDetail` objects that are associated with the current `OrderAdjustment` object. The `AdjustmentDetail` supplements the information in the `OrderAdjustment` object.
`getCurrency()`	String	The currency for all Money objects in this adjustment.

For more information on `com.bea.commerce.ebusiness.price.quote.OrderAdjustment`, refer to the *WebLogic Portal Javadoc*.

Listing 3-2 provides an example of using `getType()` to determine if the `OrderAdjusment` object is an `ORDER_DISCOUNT` type.

Listing 3-2 Using getType()

```
<%@ page
import="com.bea.commerce.ebusiness.price.quote.OrderAdjustment" %>
<%@ page
import="com.bea.commerce.ebusiness.price.quote.AdjustmentDetail"
%>

...

OrderAdjustment OrderAdjustment

AdjustmentType type = OrderAdjustment.getType()

if (type.equals(AdjustmentType.ORDER_DISCOUNT))

...
```

The AdjustmentDetail Methods

After you use the ShoppingCart methods to retrieve an `AdjustmentDetail` object, you can use `AdjustmentDetail` methods to retrieve additional information about discount or shipping information for the current JSP. Table 3-3 describes the methods that are of general interest to BEs.

Table 3-3 AdjustmentDetail Methods

Method	Returns Data of Type	Description
`getType()`	`AdjustmentType` object	One of the following types of adjustment details: ■ `ORDER_DISCOUNT` ■ `SHIPPING_BASE_PRICE` ■ `SHIPPING_DISCOUNT`
`getInitialPrice()`	Money	The order price before applying the adjustment. The `com.beasys.axiom.units.Money` class defines the Money data type.
`getEndPrice()`	Money	The order price after applying the adjustment.
`getComputation()`	String	The method of computation for the discount. For example, `50% off`.
`getReason()`	String	The reason for the discount as described in the E-Business Control Center. For example, `Spring Sale`.
`getCurrency()`	String	The currency for all `Money` objects in this adjustment detail.

For more information on `com.bea.commerce.ebusiness.price.quote.AdjustmentDetail`, refer to the *WebLogic Portal Javadoc*.

Example of a Discount for the Order

The Pricing Service determines that the entire order is eligible for a 10% discount. It applies the discount and creates an `OrderAdjustment` object that contains the following information:

- `AdjustmentType type = OrderAdjustment.getType():` ORDER_DISCOUNT
- `getBasePrice()`: $50. The order subtotal
- `getActualPrice()`: $45. The subtotal after applying the discount.
- `getAdjustmentAmount()`: $5. The order discount
- `getDetails`: One AdjustmentDetail object
- `getCurrency()`: "USD"

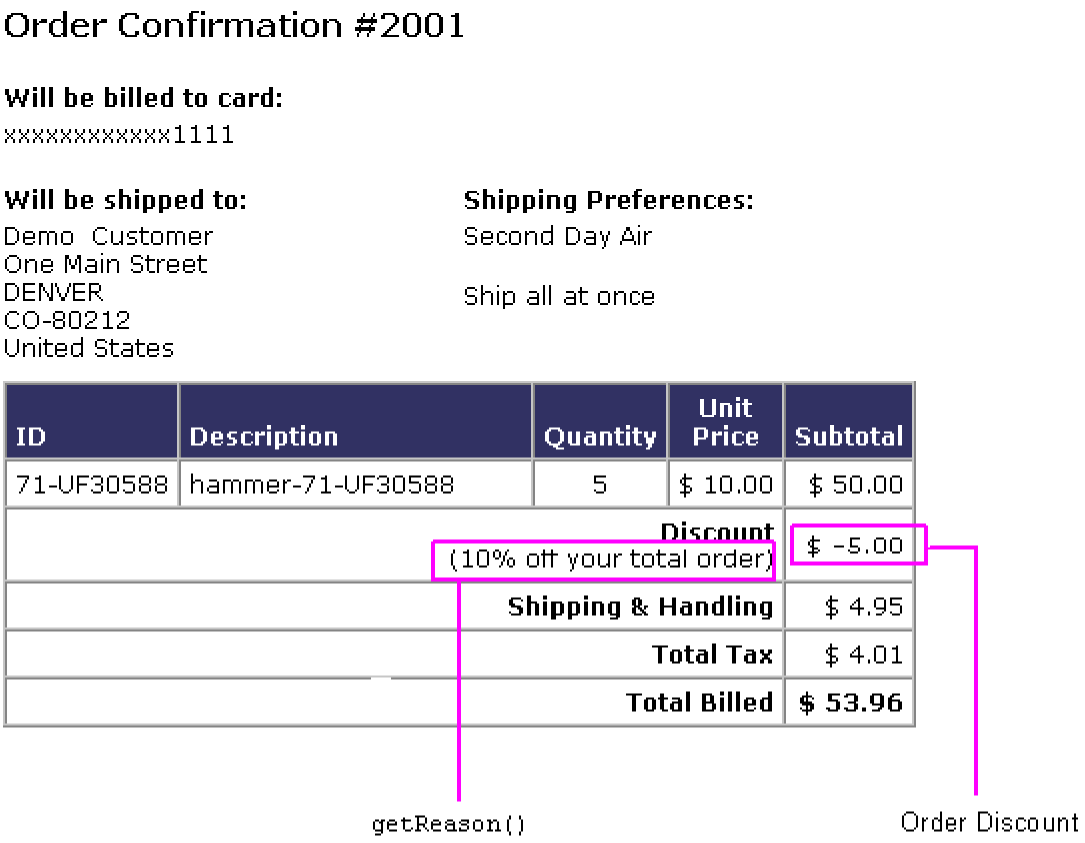
The Pricing Service also creates an `AdjustmentDetail` object to supplement the information in the `OrderAdjustment` object. The `AdjustmentDetail` object contains the following information:

- `getType()`: ORDER_DISCOUNT
- `getInitialPrice()`: $50
- `getEndPrice()`: $45
- `getComputation()`: "10% off your total order"
- `getReason()`: "First-time Buyer Discount"
- `getCurrency()`: "USD"

You can use the `getOrderDiscountPresentations()` method to retrieve the data in the `OrderAdjustment` and `AdjustmentDetails` objects. Because a shopping cart can discount either the entire order price or the shipping charges (but not both), you can use a single scriptlet to display either type of discount.

Figure 3-6 shows an example of a JSP that displays an order discount.

Figure 3-6 Order Discount Example



Order Confirmation #2001

Will be billed to card:
xxxxxxxxxxxx1111

Will be shipped to:
Demo Customer
One Main Street
DENVER
CO-80212
United States

Shipping Preferences:
Second Day Air

Ship all at once

ID	Description	Quantity	Unit Price	Subtotal
71-UF30588	hammer-71-UF30588	5	$ 10.00	$ 50.00
			Discount (10% off your total order)	$ -5.00
			Shipping & Handling	$ 4.95
			Total Tax	$ 4.01
			Total Billed	**$ 53.96**

Example of a Discount for the Shipping Charges

The Pricing Service always creates an `OrderAdjustment` object and an `AdjustmentDetails` object to describe standard shipping charges. You can use the `getShippingDiscountPresentations()` method to retrieve these objects.

Shipping Cost Without a Shipping Discount

If the shopping cart receives an order discount (and no discount for shipping charges), the shipping `OrderAdjustment` object would contain the following info:

- `getType()`: SHIPPING
- `getBasePrice()`: $50. The order subtotal
- `getActualPrice()`: $54.95. The new subtotal after calculating shipping charges
- `getAdjustmentAmount()`: -$4.95. The shipping charges. (A negative amount raises the price.)
- `getDetails`: List of 1 AdjustmentDetail object
- `getCurrency()`: "USD"

If the shopping cart receives an order discount (and no discount for shipping charges), the `AdjustmentDetail` object contains the following:

- `getType()`: SHIPPING_BASE_PRICE
- `getInitialPrice()`: $50
- `getEndPrice()`: $54.95
- `getComputation()`: ""   (empty string)
- `getReason()`: "Base Shipping Charges"
- `getCurrency()`: "USD"

Shipping Discount Applied to the Shipping Cost

If the shopping cart receives a shipping discount (instead of an order discount), the shopping cart method `getOrderDiscountPresentations()` would return a null value.

The `getShippingDiscountPresentations()` would return a single `OrderAdjustment` object and two `AdjustmentDetail` objects: one that describes the standard shipping charges and another that describes the shipping discount.

The `OrderAdjustment` object would contain the following information:

- `getType()`: SHIPPING
- `getBasePrice()`: $50
- `getActualPrice()`: $50. Order subtotal after calculating the shipping discount.
- `getAdjustmentAmount()`: $0. The shipping cost after applying all shipping adjustments. (A negative value raises the price.)
- `getDetails`: List of 2 AdjustmentDetail objects.
- `getCurrency()`: "USD".

The first `AdjustmentDetail` object contains the following:

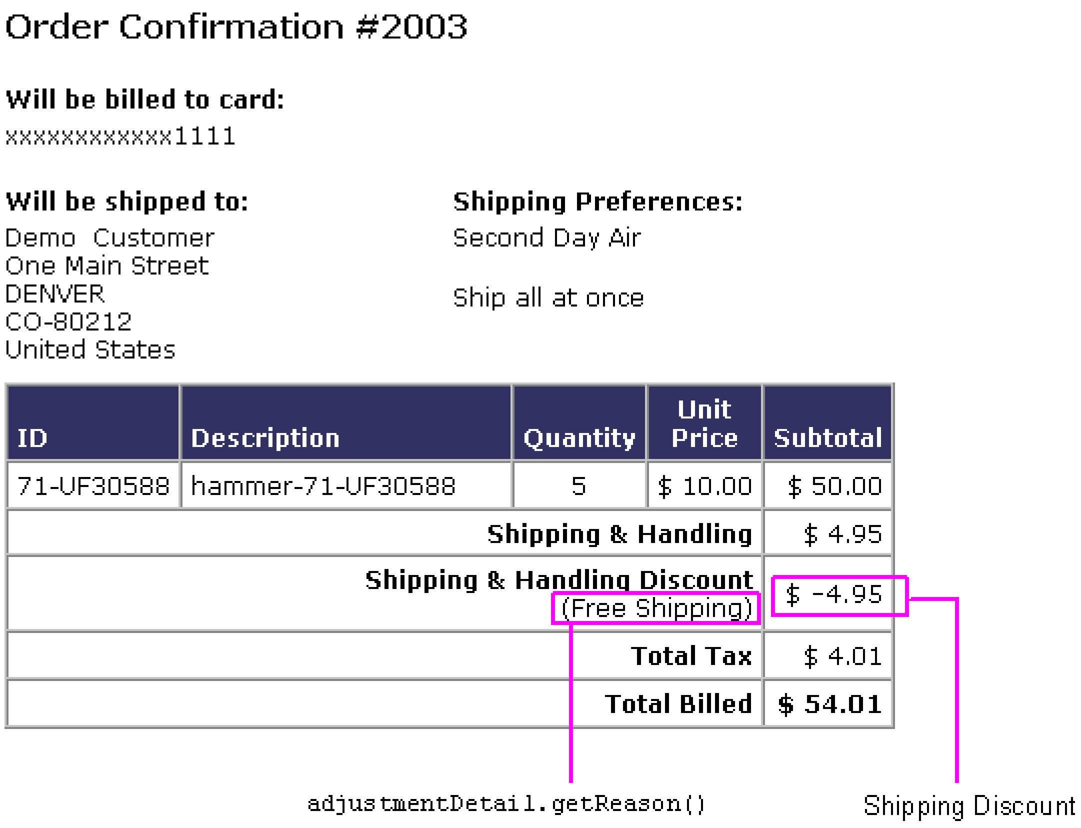
- `getType()`: SHIPPING_BASE_PRICE
- `getInitialPrice()`: $50
- `getEndPrice()`: $54.95
- `getComputation()`: ""
- `getReason()`: "Base Shipping Charges"
- `getCurrency()`: "USD"

The second `AdjustmentDetail` object contains the following:

- `getType()`: SHIPPING_DISCOUNT
- `getInitialPrice()`: $54.95
- `getEndPrice()`: $50
- `getComputation()`: "100% off shipping charges."
- `getReason()`: "Free Shipping"
- `getCurrency()`: "USD"

Figure 3-7 shows a JSP that displays a shipping discount.

Figure 3-7 Shipping Discount Example



Order Confirmation #2003

Will be billed to card:
xxxxxxxxxxxx1111

Will be shipped to:
Demo Customer
One Main Street
DENVER
CO-80212
United States

Shipping Preferences:
Second Day Air

Ship all at once

ID	Description	Quantity	Unit Price	Subtotal
71-UF30588	hammer-71-UF30588	5	$ 10.00	$ 50.00
			Shipping & Handling	$ 4.95
			Shipping & Handling Discount (Free Shipping)	$ -4.95
			Total Tax	$ 4.01
			Total Billed	**$ 54.01**

CHAPTER

4 Setting Up and Sending E-mail for Campaigns

WebLogic Portal provides a default Mail Service, based on JavaMail, to process e-mail requests from scenario actions.

This topic includes the following sections:

- How Campaigns Use the Mail Service
- Configuring E-Mail Properties for Campaigns
- Creating E-mail JSPs
- Sending Bulk Mail

Instead of the WebLogic Portal Mail Service, you can use a third-party mail service.

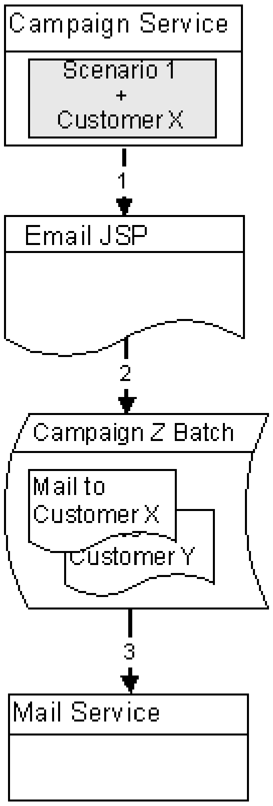
How Campaigns Use the Mail Service

Business Analysts (BAs) can specify that a scenario within a campaign sends an e-mail to a customer. For example, when a customer buys a flashlight, a scenario can send a an e-mail that contains special offers for batteries.

When BAs create scenarios, they select a JSP file that contains the e-mail content. When a customer triggers the scenario action, the following process occurs:

1. The Campaign Service uses internal HTTP to request the JSP URI that the scenario action specifies. In the request, it passes parameters to identify the name of the scenario and the identity of the customer who triggered the scenario action.
2. The JSP invokes any JSP tags that it contains, uses MIME types to encode non-ASCII output, and stores its output in the WebLogic Portal data repository. The data repository organizes the e-mails into batches; one batch for each campaign.
3. You issue a command to the Mail Service that specifies which batch of messages you want to send. The Mail Service uses the JavaMail API to send the messages.

Figure 4-1 The Mail Service for Campaigns



Configuring E-Mail Properties for Campaigns

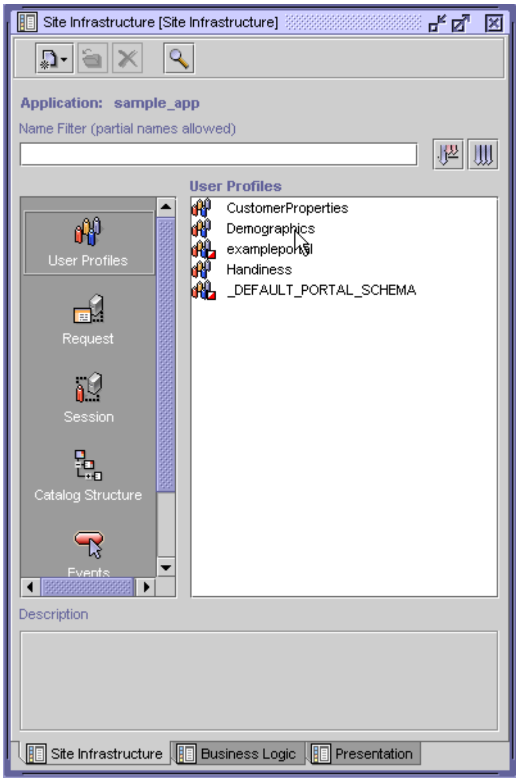
Before a campaign can send e-mail, you must configure properties that the Campaign Service uses to send and receive mail. In a clustered environment, WebLogic Server propagates these properties to each node in the cluster.

To configure mail-related properties, do the following:

1. From the E-Business Control Center, open your application.
2. In the E-Business Control Center Explorer window, click the Site Infrastructure tab.
3. Click User Profiles and find the following (see Figure 4-2):
   - The name of the property set and the property that defines customer e-mail addresses.
   - The name of the property set and the property that records a customer's preference for receiving campaign-related e-mail. The reference applications store this preference in the `Demographics` property set in the `Email_Opt_In` property.

   For information about property sets and properties, refer to "Creating and Managing Property Sets" in the *Guide to Building Personalized Applications*.

Figure 4-2 Find Names of User Properties



4. Start your server and access the WebLogic Server Administration Console for the domain. For more information, refer to “The WebLogic Server Administration Console” in the *WebLogic Portal Architectural Overview*.

5. In the left pane of the WebLogic Server Administration Console, click Deployments → Applications → *myApplication* → Service Configuration → Campaign Service.

6. On the Campaign Service Page, click the Mail Action tab.

7. On the Mail Action tab, enter the following values:

In this box...	**Enter this value...**
Default From Email Address	The default address that receives any replies from e-mail that the campaign sends. In a standard mail header, this is the From address. Each campaign scenario can specify its own From address that overrides this default property.
Email Address Property Name	The name of the property that contains customer e-mail addresses.
Property Set Name Containing Email Address Property	The name of the property set that contains customer e-mail properties.
Email Opt In Property Name	The name of the property that specifies whether customers want to receive campaign-related email. The reference applications store this preference in the `Demographics` property set in the `Email_Opt_In` property.
Property Set Name Containing Opt In Property	The name of the property set that contains the customer's opt-in property.

8. Click Apply. All e-mail that the Campaign Service generates will now use these settings.

9. To configure the default SMTP host name for the Mail Service, in the left pane, click Mail Service. Any changes that you make on the Mail Service page affects all e-mails (whether or not they are generated by the Campaign Service) that you send using WebLogic Portal.

Creating E-mail JSPs

The Mail Service requires that you place the content and formatting of your e-mails in a JSP file. In this JSP, you can use any of the JSP tags and APIs that are available to other JSPs in WebLogic Portal.

This section describes the following:

- E-mail Parameters
- Disabling Session Generation
- Sample E-mail JSP
- Location of E-Mail JSPs

E-mail Parameters

When a scenario action requests an e-mail JSP, it passes a `userId` parameter, which specifies the login name of the customer who triggered the scenario action. With the `request.getParameter` method, you can retrieve the user ID and pass it to JSP tags in the e-mail JSP.

In addition, the scenario passes the following parameters (you can also pass these parameters to JSP tags in the e-mail JSP):

- `scenarioId`, which specifies the ID of the scenario that triggered the e-mail.
- `scenarioName`, which specifies the name of the scenario that triggered the e-mail.
- `containerId`, which specifies the ID of the campaign to which the scenario belongs.
- `containerName`, which specifies the name of the campaign to which the scenario belongs.

Disabling Session Generation

The Java class that the Campaign Service uses to generate email from a JSP, `InternalRequestDispatcher`, also generates an `HTTPSession` object. Usually, generating this `HTTPSession` from an email JSP is extraneous because your application already generates an `HTTPSession` object when a customer accesses your site.

To disable the generation of an extraneous `HTTPSession`, add the following directive to the beginning of the JSPs that you use to generate email for campaigns:

```
<%@ page session="false" %>
```

Adding this directive is necessary only if your application generates HTTPSession objects when customers access your site (or log in) and only for email that is generated via the `InternalRequestDispatcher`.

Sample E-mail JSP

Listing 4-1 shows the e-mail JSP that is part of the wlcs sample Web application. The file is located at
`PORTAL_HOME/applications/wlcsApp/wlcs/campaigns/emails/Sample1.jsp`

Listing 4-1 Sample E-mail JSP

```
<%@ page session="false" %>

<%@ page contentType="text/plain" %>

(This sample e-mail was automatically sent out as part of a sample
campaign that you triggered while registering as a new user on the
BEA Commerce Templates.)

-------------------------------

Hello <%= request.getParameter("userId") %>,

Thank you for taking the time to become a registered member of our
site. We hope you took advantage of your $10 discount on a purchase
of $50 or more after you registered!

In addition, your registration entitles you to premium services
including:

       **Special "Members Only" discounts
       **Advance notice of new product releases
       **A personalized customer experience customized to your
          specific interests

Thanks again for becoming a registered member.

Best Regards
```

Location of E-Mail JSPs

You must save e-mail JSPs in a specific directory within a Web application so that BAs can use the E-Business Control Center to browse and select the e-mail for a campaign.

By default, the directory is `*myApp/myWebApp/*campaigns/emails`.

For example, the wlcs application provides a sample e-mail in the following directory: `PORTAL_HOME/applications/wlcsApp/wlcs/campaigns/emails/Sample1.jsp`

To choose this e-mail as part of a scenario action, a BA opens the wlcsApp application in the E-Business Control Center. Then, while creating an E-mail action, the BA can browse through all e-mail JSPs that have been saved in `PORTAL_HOME/applications/wlcsApp/wlcs/campaigns/emails` and select `Sample1.jsp` for the scenario.

To change the default location in which you save JSP e-mail, do the following:

1. From the WebLogic Server Administration Console, in the left pane, click Deployments → Applications → *myApplication* → Service Configuration→ Campaign Service.
2. On the Campaign Service page, click the Configuration tab.
3. In the Base Directory for Email Browsing box, enter a pathname that is relative to the root directory of a Web application.

Sending Bulk Mail

You must periodically use a command to send the batches e-mail that the JSPs store in the WebLogic Portal data repository. You can also use `cron` or any other scheduler that your operating system supports to issue the send-mail command.

On Windows, the send-mail command is in a `.bat` file wrapper script; on UNIX it is in a `.sh` file. This section refers to the `.bat` file. UNIX users should substitute `.sh` for `.bat`.

This section includes the following subsections:

- Sending Mail from a Remote Host or in a Clustered Environment
- To Send Bulk E-mail
- To Delete E-mail Batches
- Scheduling Bulk E-mail Delivery
- Mailmanager Command Reference

Sending Mail from a Remote Host or in a Clustered Environment

The send-mail wrapper script specifies the name and listen port of the WebLogic Portal host that processes the send-mail request. By default, the wrapper script specifies `localhost:7501` for the hostname and listen port. However, `localhost:7501` is valid only when you run the script while logged in to a WebLogic Portal host in a single-node environment (and only if you did not modify the default listen port).

Before you use the send-mail script from any other configuration, you must modify the script. This section describes the following tasks:

- Modifying the Send-Mail Script to Work from a Remote Host
- Modifying the Send-Mail Script to Work in a Clustered Environment

Modifying the Send-Mail Script to Work from a Remote Host

If you want to run the send-mail script from a remote host (that is, a computer that is not a WebLogic Portal host), do the following:

1. Open the following file in a text editor:
   `PORTAL_HOME\bin\win32\mailmanager.bat` (Windows)
   `PORTAL_HOME/bin/unix/mailmanager.sh` (UNIX)
2. In the `mailmanager` script, in the `SET HOST=` line, replace `localhost` with the name of a WebLogic Portal host.
3. If the host uses a listen port other than `7501`, in the `SET PORT=` line, replace `7501` with the correct listen port.
4. Save the `mailmanager` script.

Modifying the Send-Mail Script to Work in a Clustered Environment

If you work in a clustered environment, you must modify the send-mail wrapper script to specify the name of a host in the cluster. The default `localhost` value is not valid for the Mail Service in a clustered environment.

To use the send-mail script in a clustered environment, **do the following on each host from which you want to run the script**:

1. Open the following file in a text editor:
   `PORTAL_HOME\bin\win32\mailmanager.bat` (Windows)
   `PORTAL_HOME/bin/unix/mailmanager.sh` (UNIX)
2. In the `mailmanager` script, in the `SET HOST=` line, replace `localhost` with the name of a WebLogic Portal host. Because each host in a cluster can access the data repository that stores the e-mail messages, you can specify the name of any host in the cluster.
3. If the host uses a listen port other than `7501`, in the `SET PORT=` line, replace `7501` with the correct listen port.
4. Save the `mailmanager` script.

To Send Bulk E-mail

To send bulk e-mail, do the following from a shell that is logged in to a WebLogic Portal host:

1. To determine the names and contents of the e-mail batches in the data repository, enter the following command:

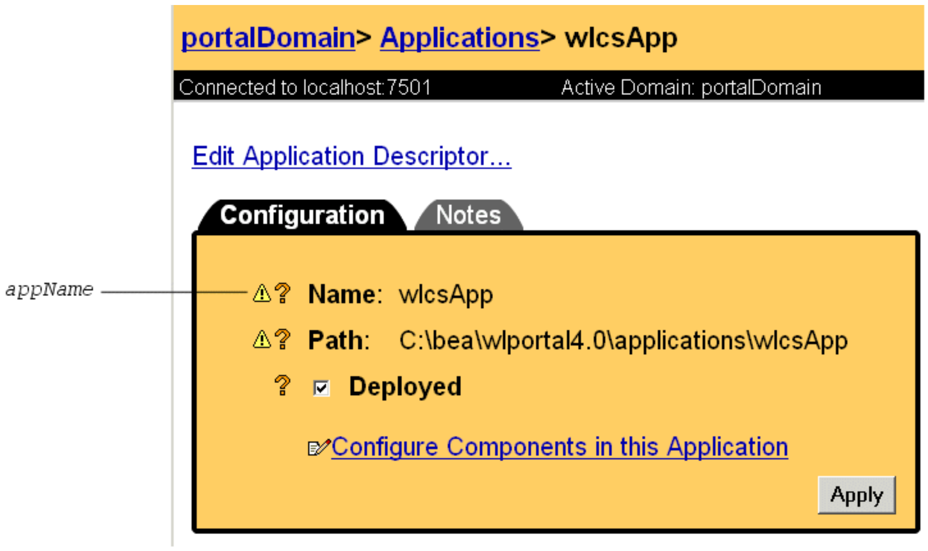
   `mailmanager.bat` *`appName`* `list` (Windows)

   where *`appName`* is the name of the enterprise application that generated the e-mail batch. (See Figure 4-3.) The command prints to standard out. You can use shell commands to direct the output to files.

2. To send a batch and remove it from the data repository, enter the following command:

   `mailmanager.bat` *`appName`* `send-delete` *`batch-name`*

Figure 4-3 Application Name



portalDomain> Applications> wlcsApp

Connected to localhost:7501 Active Domain: portalDomain

Edit Application Descriptor...

Name: wlcsApp

Path: C:\bea\wlportal4.0\applications\wlcsApp

☑ Deployed

Configure Components in this Application

To Delete E-mail Batches

You can delete e-mail batches as you send them (as described in To Send Bulk E-mail). You can also do the following to delete e-mail batches:

1. To determine the names and contents of the e-mail batches in the data repository, enter the following command:

   ```
   mailmanager.bat appName list
   ```

   where *`appName`* is the name of the enterprise application that generated the e-mail batch. (See Figure 4-3.) The command prints to standard out. You can use shell commands to direct the output to files.

2. To delete a batch, enter the following command:

   ```
   mailmanager.bat appName delete batch-name
   ```

Scheduling Bulk E-mail Delivery

You can use a scheduling utility to send the e-mail batches in the data repository. Because you must specify the name of a batch when you use the `mailmanager` command to send mail, you must schedule sending mail for each campaign scenario separately. The name of a batch corresponds to the scenario’s container ID. For information about the container ID, refer to “E-mail Parameters” on page 4-68.

For information in using a scheduling utility, refer to the documentation for your operating system.

Mailmanager Command Reference

The `mailmanager` command is a wrapper script that uses `java com.bea.p13n.mail.MailManager`. The command syntax is as follows:

`mailmanager.bat [` *`appName`* `] [ list | send | send-delete | delete ] [` *`batch-name`* `]` (`mailmanager.sh` on UNIX)

If you specify only the *`appName`* arguments `mailmanager` prints to standard output the names all e-mail batches in the application and the number of e-mails in each batch.

Use the command arguments as follows:

- *`appName`*

  The name of the enterprise application that generated the e-mail batch. (See Figure 4-3.)

- `list`

  Prints to standard output the names of all e-mail batches in the data repository and the number of e-mails in each batch.

- `list` *`batch-name`*

  Prints to standard output the subject and recipients of all e-mail in the batch that you specify.

- `send` *`batch-name`*

  Sends all e-mails in the batch that you specify.

- `send-delete` *`batch-name`*

  Sends all e-mails in the batch that you specify and then deletes the batch from the data repository.

- `delete` *`batch-name`*

  Deletes e-mails in the batch that you specify.

- *`batch-name`*

  The name of a batch that `mailmanager list` returns. This argument does not support wildcards.

Command Examples

To list all available batches, enter the following command:

```
mailmanager.bat list
```

To list the contents of a batch named `/campaigns/campaign1.cam` that the wlcsApp application generated, enter the following command:

```
mailmanager.bat wlcsApp list /campaigns/campaign1.cam
```

To send the `campaign1.cam` batch and delete it afterwards, enter the following command:

```
mailmanager.bat wlcsApp send-delete /campaigns/campaign1.cam
```

To delete the `campaign1.cam` batch, enter the following command:

```
mailmanager.bat wlcsApp delete /campaigns/campaign1.cam
```

CHAPTER

5 Campaign Manager Database Schema

This topic describes the database schema for the Campaign services. Understanding this schema will be helpful to those who may be customizing or extending the technologies provided in the product.

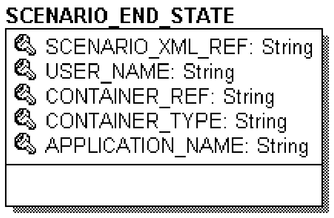
This topic includes the following sections:

- The Entity-Relation Diagram
- List of Tables Composing the BEA Campaign Manager
- The Campaign Manager Data Dictionary
- The SQL Scripts Used to Create the Database
- Defined Constraints

The Entity-Relation Diagram

Figure 5-1 shows the Entity-Relation diagram for the E-Business Control Center for the Campaign services database. See the subsequent sections in this chapter for information about the data type syntax.

Figure 5-1 Entity-Relation Diagram for Campaign Manager Database Tables



List of Tables Composing the BEA Campaign Manager

The BEA Campaign Manager is composed of the following table:

Campaign and Scenarios

- The SCENARIO_END_STATE Database Table

The Campaign Manager Data Dictionary

At this time, there is only one database table pertaining to the Campaign Manager.

See also the chapter “The WebLogic Personalization Server Database Schema” in the *Guide to Building Personalized Applications*.

The SCENARIO_END_STATE Database Table

Table 5-1 describes the metadata for the E-Business Control Center SCENARIO_END_STATE table. This table identifies when a user is no longer eligible to participate in a particular scenario.

The Primary Keys are `SCENARIO_XML_REF`, `USER_NAME`, `CONTAINER_REF`, `CONTAINER_TYPE` and `APPLICATION_NAME`.

Table 5-1 SCENARIO_END_STATE Table Metadata

Column Name	Data Type	Null Value	Description and Recommendations
`SCENARIO_XML_REF`	`VARCHAR(20)`	`NOT NULL`	PK—The identifier for the XML-based scenario definition file.
`USER_NAME`	`VARCHAR(200)`	`NOT NULL`	PK—the user `ID`. (FK to `WLCS_USER.IDENTIFIER`)
`CONTAINER_REF`	`VARCHAR(254)`	`NOT NULL`	PK—the campaign unique identifier. (FK to `CAMPAIGN.CAMPAIGN_UID`)
`CONTAINER_TYPE`	`VARCHAR(50)`	`NOT NULL`	PK—At this time this column will always hold the string 'Campaign'.
`APPLICATION_NAME`	`VARCHAR(100)`	`NOT NULL`	PK—The deployed J2EE application name. This should match the name in the WebLogic Server console.)

The SQL Scripts Used to Create the Database

The database schemas for WebLogic Portal and WebLogic Personalization Server are all created by executing the `create_all` script for the target database environment.

Scripts

Regardless of your database, execute one of the following to generate the necessary database objects for the modules desired (WebLogic Portal, WebLogic Personalization Server, Commerce services, Campaign services and Sample Portal):

- `PORTAL_HOME\db\create_all.bat` (Windows)
- `PORTAL_HOME/db/create_all.sh` (UNIX)

The following are the various directories underneath `WL_COMMERCE_HOME/db` (as seen in a UNIX environment):

`PORTAL_HOME/db/cloudscape/351`

`PORTAL_HOME/db/oracle/817`

Note: In this documentation,`PORTAL_HOME` is used to designate the directory where the product is installed.

Each of the databases supported have the same number of scripts in each of their sub-directories. The scripts are listed and described in Table 5-2 below.

Table 5-2 The Scripts Supporting the Databases

Script Name	Description
`create_all.bat`	Windows script used to connect to the database and create the necessary database objects for the modules desired (e.g., WebLogic Portal, WebLogic Personalization Server, Commerce services, Campaign services and Sample Portal)

Table 5-2 The Scripts Supporting the Databases (Continued)

Script Name	Description
`create_all.sh`	Unix script used to connect to the database and create the necessary database objects for the modules desired (e.g., WebLogic Portal, WebLogic Personalization Server, Commerce services, Campaign services and Sample Portal)
`campaign_create_fkeys.sql`	SQL script used to create all foreign keys associated with the Campaign services.
`campaign_create_indexes.sql`	SQL script used to create all indexes associated with the Campaign services.
`campaign_create_tables.sql`	SQL script used to create all tables associated with the Campaign services.
`campaign_create_triggers.sql`	SQL script used to create all database triggers associated with the Campaign services.
`campaign_create_views.sql`	SQL script used to create all views associated with the Campaign services.
`campaign_drop_constraints.sql`	SQL script used to drop all constraints (other than foreign keys) associated with the Campaign services.
`campaign_drop_fkeys.sql`	SQL script used to drop all foreign key constraints associated with the Campaign services.
`campaign_drop_indexes.sql`	SQL script used to drop all indexes associated with the Campaign services.
`campaign_drop_tables.sql`	SQL script used to drop all tables associated with the Campaign services.
`campaign_drop_views.sql`	SQL script used to drop all views associated with the Campaign services.
`p13n_create_fkeys.sql`	SQL script used to create all foreign keys associated with the WebLogic Personalization Server.
`p13n_create_indexes.sql`	SQL script used to create all indexes associated with the WebLogic Personalization Server.
`p13n_create_tables.sql`	SQL script used to create all tables associated with the WebLogic Personalization Server.
`p13n_create_triggers.sql`	SQL script used to create all database triggers associated with the WebLogic Personalization Server.

Table 5-2 The Scripts Supporting the Databases (Continued)

Script Name	Description
`p13n_create_views.sql`	SQL script used to create all views associated with the WebLogic Personalization Server.
`p13n_drop_constraints.sql`	SQL script used to drop all constraints (other than foreign keys) associated with the WebLogic Personalization Server.
`p13n_drop_fkeys.sql`	SQL script used to drop all foreign key constraints associated with the WebLogic Personalization Server.
`p13n_drop_indexes.sql`	SQL script used to drop all indexes associated with the WebLogic Personalization Server.
`p13n_drop_tables.sql`	SQL script used to drop all tables associated with the WebLogic Personalization Server.
`p13n_drop_views.sql`	SQL script used to drop all views associated with the WebLogic Personalization Server.
`portal_create_fkeys.sql`	SQL script used to create all foreign keys associated with the WebLogic Portal.
`portal_create_indexes.sql`	SQL script used to create all indexes associated with the WebLogic Portal.
`portal_create_tables.sql`	SQL script used to create all tables associated with the WebLogic Portal.
`portal_create_triggers.sql`	SQL script used to create all database triggers associated with the WebLogic Portal.
`portal_create_views.sql`	SQL script used to create all views associated with the WebLogic Portal.
`portal_drop_constraints.sql`	SQL script used to drop all constraints (other than foreign keys) associated with the WebLogic Portal.
`portal_drop_fkeys.sql`	SQL script used to drop all foreign key constraints associated with the WebLogic Portal.
`portal_drop_indexes.sql`	SQL script used to drop all indexes associated with the WebLogic Portal.
`portal_drop_tables.sql`	SQL script used to drop all tables associated with the WebLogic Portal.
`portal_drop_views.sql`	SQL script used to drop all views associated with the WebLogic Portal.

Table 5-2 The Scripts Supporting the Databases (Continued)

Script Name	Description
`sample_portal_create_fkeys.sql`	SQL script used to create all foreign keys associated with the Sample Portal.
`sample_portal_create_indexes.sql`	SQL script used to create all indexes associated with the Sample Portal.
`sample_portal_create_tables.sql`	SQL script used to create all tables associated with the Sample Portal.
`sample_portal_create_triggers.sql`	SQL script used to create all database triggers associated with the Sample Portal.
`sample_portal_create_views.sql`	SQL script used to create all views associated with the Sample Portal.
`sample_portal_drop_constraints.sql`	SQL script used to drop all constraints (other than foreign keys) associated with the Sample Portal.
`sample_portal_drop_fkeys.sql`	SQL script used to drop all foreign key constraints associated with the Sample Portal.
`sample_portal_drop_indexes.sql`	SQL script used to drop all indexes associated with the Sample Portal.
`sample_portal_drop_tables.sql`	SQL script used to drop all tables associated with the Sample Portal.
`sample_portal_drop_views.sql`	SQL script used to drop all views associated with the Sample Portal.
`wlcs_create_fkeys.sql`	SQL script used to create all foreign keys associated with the Commerce services.
`wlcs_create_indexes.sql`	SQL script used to create all indexes associated with the Commerce services.
`wlcs_create_tables.sql`	SQL script used to create all tables associated with the Commerce services.
`wlcs_create_triggers.sql`	SQL script used to create all database triggers associated with the Commerce services.
`wlcs_create_views.sql`	SQL script used to create all views associated with the Commerce services.
`wlcs_drop_constraints.sql`	SQL script used to drop all constraints (other than foreign keys) associated with the Commerce services.

Table 5-2 The Scripts Supporting the Databases (Continued)

Script Name	Description
`wlcs_drop_fkeys.sql`	SQL script used to drop all foreign key constraints associated with the Commerce services.
`wlcs_drop_indexes.sql`	SQL script used to drop all indexes associated with the Commerce services.
`wlcs_drop_tables.sql`	SQL script used to drop all tables associated with the Commerce services.
`wlcs_drop_views.sql`	SQL script used to drop all views associated with the Commerce services.

Defined Constraints

There are not constraints associated with this part of the schema.

.

Index