BEA Campaign Manager for WebLogic

Guide to Developing Campaign Infrastructure

BEA Campaign Manager for WebLogic 1.1
Document Edition 3.5.1
June 2001

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Guide to Developing Campaign Infrastructure

Document Edition	Date	Software Version
3.5.1	June 2001	BEA Campaign Manager for WebLogic 1.1, Service Pack 1
3.5	April 2001	BEA Campaign Manager for WebLogic 1.1

Contents

3. Setting Up JSP Tags and Scriptlets for Campaigns

4. Setting Up and Sending E-mail for Campaigns

5. Campaign Manager Database Schema

Index

About This Document

This document describes setting up infrastructure for the features that the Campaign Service uses. The Campaign Service is available only with BEA Campaign Manager for WebLogic™. This document 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 BEA Campaign Manager package. 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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server documentation Home page on the e-docs Web site (and also on the documentation CD). 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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server 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

Because Campaign Manager for WebLogic adds to the services of WebLogic Commerce Server™ and WebLogic Personalization Server™, you must set up infrastructure for WebLogic Commerce Server and WebLogic Personalization Server as well. For example, you must create JSP tags to retrieve items from the product catalog and add the items to the shopping cart.

For information on setting up WebLogic Commerce Server and WebLogic Personalization Server infrastructure, 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 Creating Portals and Portlets*
- *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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server 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 Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server **Product Version: 3.2** release.

If you have any questions about this version of BEA Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, or if you have problems installing and running BEA Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server, 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.

Convention	Item

`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. *Example*s: LPT1 SIGNON OR
`{ }`	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.

Convention	Item

. . .	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 Commerce Business Engineer (CBE). A BA develops the strategy and goals for individual campaigns, and uses the BEA E-Business Control Center to run and evaluate them. A CBE 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 CBE perspective. It includes the following sections:

- What Is a Campaign?
- How Data Flows in a Campaign
- Workflow for Developing Campaign Infrastructure
- Viewing Campaign Messages in WebLogic Server Administration Console

What Is a Campaign?

A campaign coordinates several WebLogic Personalization Server, WebLogic Commerce Server, and Campaign Manager for WebLogic 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.

Campaigns coordinate the following services:

- Events and Behavior Tracking Services, which identify how a customer interacts with your site. By default Campaign Manager for WebLogic 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. 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 Campaign Manager for WebLogic 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. A BA creates a placeholder in the E-Business Control Center, and you create a placeholder JSP tag in the location in which the BA wants to display ads. 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: In addition to events that the Event Service detects, a date or range of dates can initiate actions in a campaign.

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.

   Note: For a description of the events that belong to these event types, refer to "Event Details" under "Overview of Events and Behavior Tracking" in the *Guide to Events and Behavior Tracking*.

4. If the event is one of the types described in the previous step, the Campaign Listener notifies the Campaign Service.

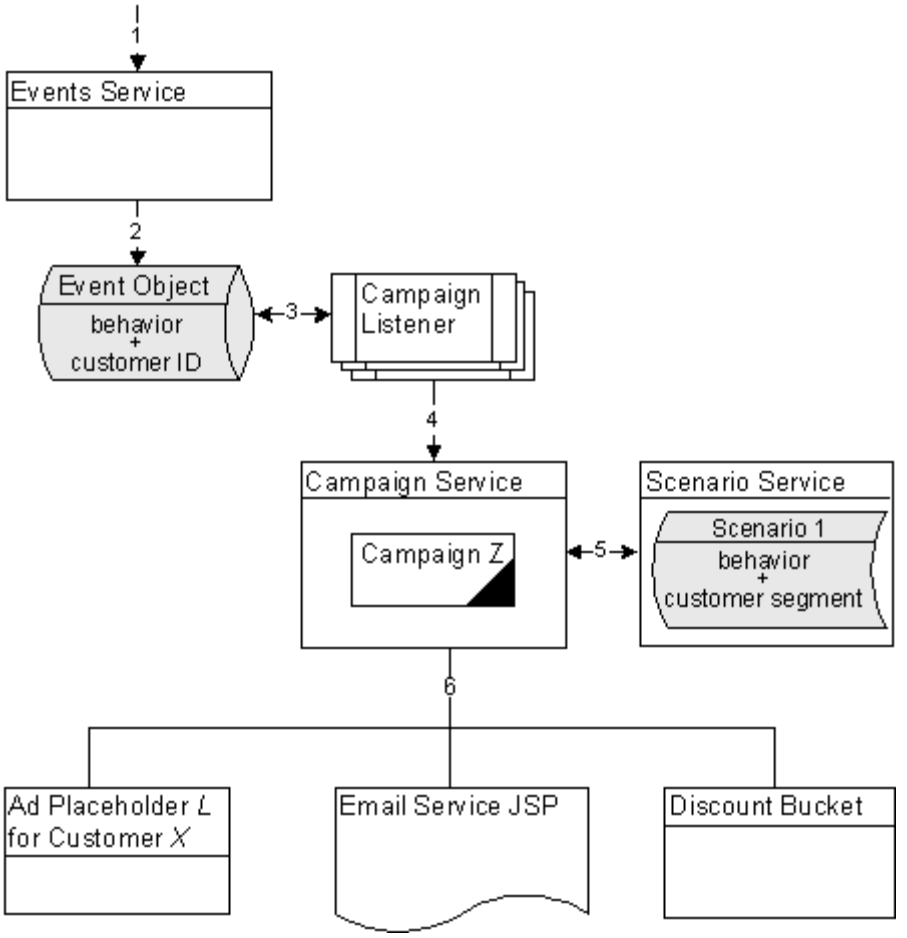
5. For each active campaign, the Campaign Service attempts to match the event with a scenario. For example, customer Pat Gomes is a Gold Customer who added a handsaw to the shopping cart. Scenario 1 in Campaign Z specifies actions for Gold Customers who add handsaws to the shopping cart.

6. For any scenario that matches the event, the Scenario Service finds an action to initiate. 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.

   Then the Campaign Service initiates the action. 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:

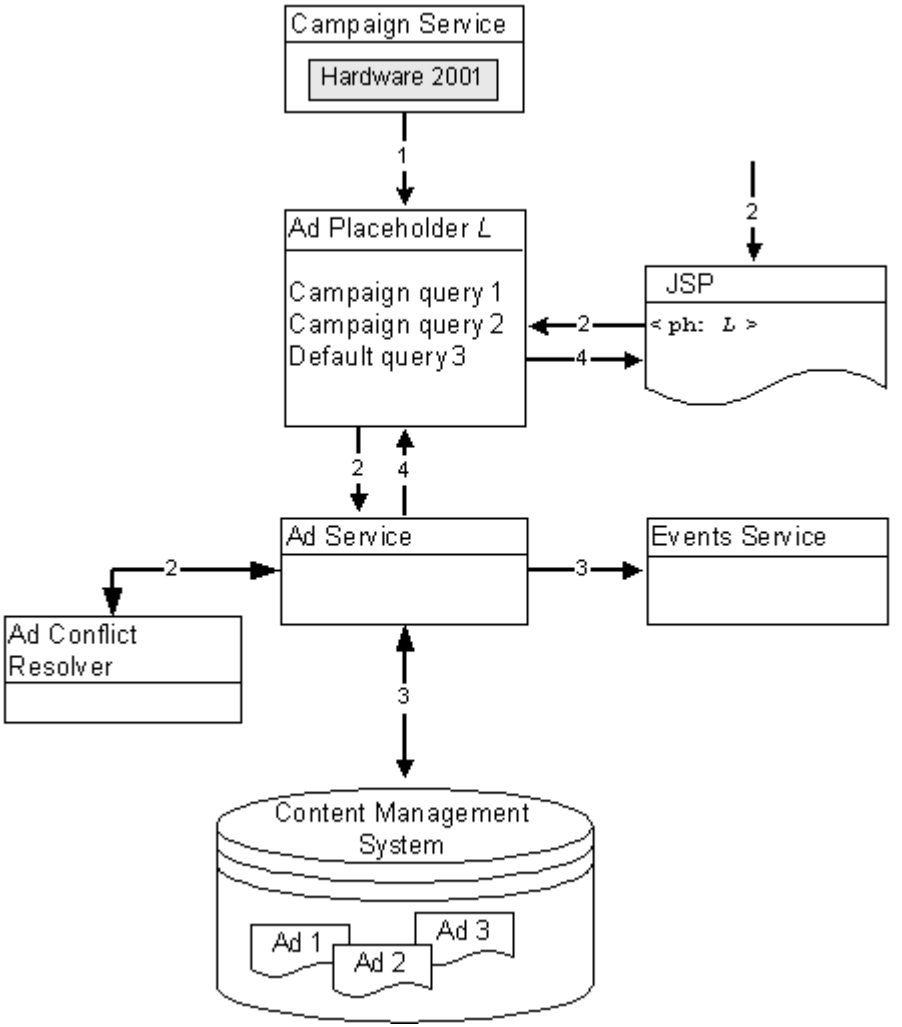
   a. It forwards the query to the content management system. If the query returns more than one ad, the ad placeholder uses the `adWeight` attribute of each ad to determine which one to retrieve.

   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.

   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* and "Campaign Service Properties" under "The Server Configuration" in the *Deployment Guide*.
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



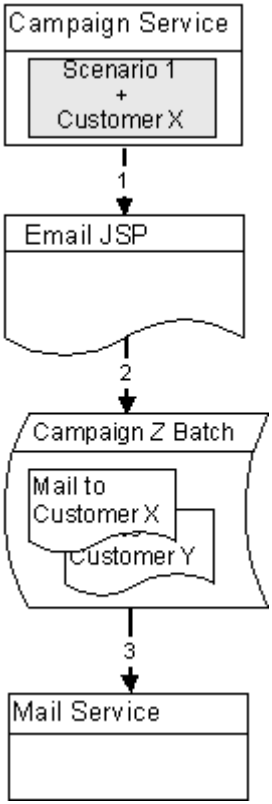
How Campaigns Use the Mail Service

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, they provide the URI of 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 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 Campaign Manager for WebLogic 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

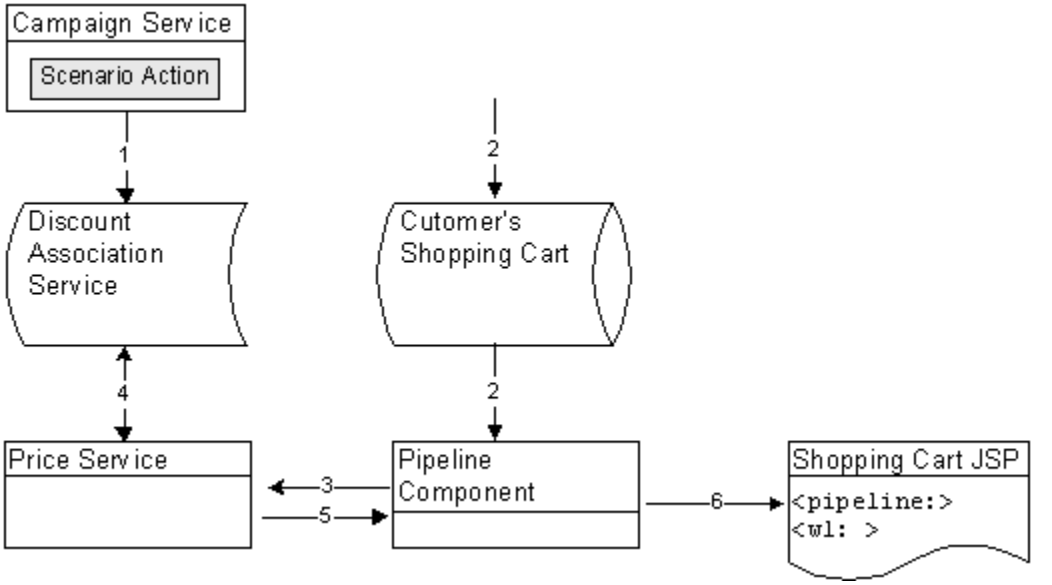
A BA creates discounts while defining actions for a scenario. When an event triggers a scenario to activate a discount the following process occurs:

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
   - The number of orders to which this discount has been applied

   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**. Campaign Manager for WebLogic 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 "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.

3. **Create JSP Tags**. On any JSP that generates or reacts to events for a campaign, you must create a `<um:getProfile>` JSP tag to retrieve the customer's profile. A customer profile is a key piece of information for determining whether a campaign scenario applies to a specific event. You may have already added this JSP tag to support other personalization or behavior tracking features.

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

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 Campaign Manager for WebLogic, you can set up an integration with a third-party e-mail service.

5. **Send Bulk E-mails**. The Campaign Manager for WebLogic 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.

Viewing Campaign Messages in WebLogic Server Administration Console

To provide an audit trail of their activities, campaigns send messages to the WebLogic Server messaging system. Campaigns also send error and warning messages to this system.

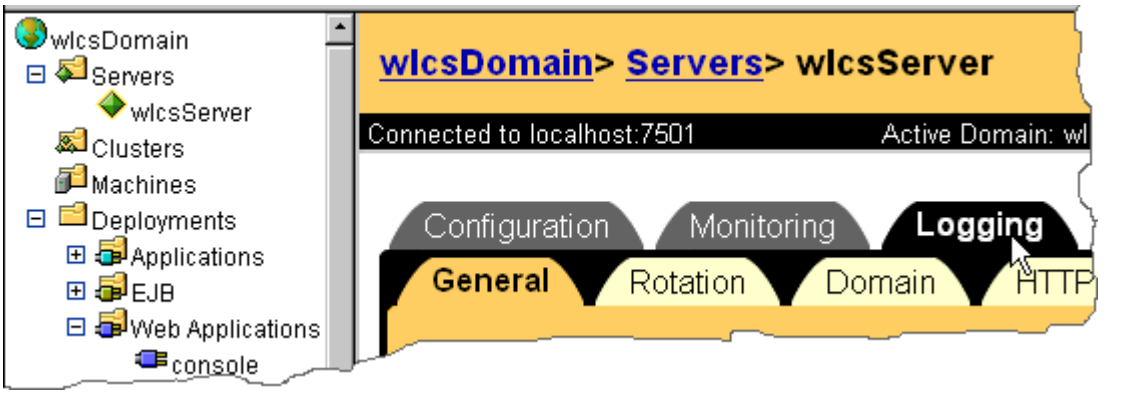
You can view these messages from the WebLogic Server Administration Console. To view only messages that campaigns generate, use the following string to filter messages:

```
Campaign and Scenarios
```

To view campaign messages from the WebLogic Server Administration Console, do the following:

1. Start the Administration Console. For more information, refer to "The Server Configuration," in the *Deployment Guide*.
2. In Administration Console, in the left pane, click Servers → wlcsServer.
3. On the wlcsServer page, click the Logging tab. (See Figure 1-5.)

Figure 1-5 The Logging Tab



4. On the Logging tab, click View Server Log.

   The Administration Console displays the Log page, which contains all Campaign Manager for WebLogic messages.

5. To view only messages that campaigns generate, on the Log page click Search and Customize.

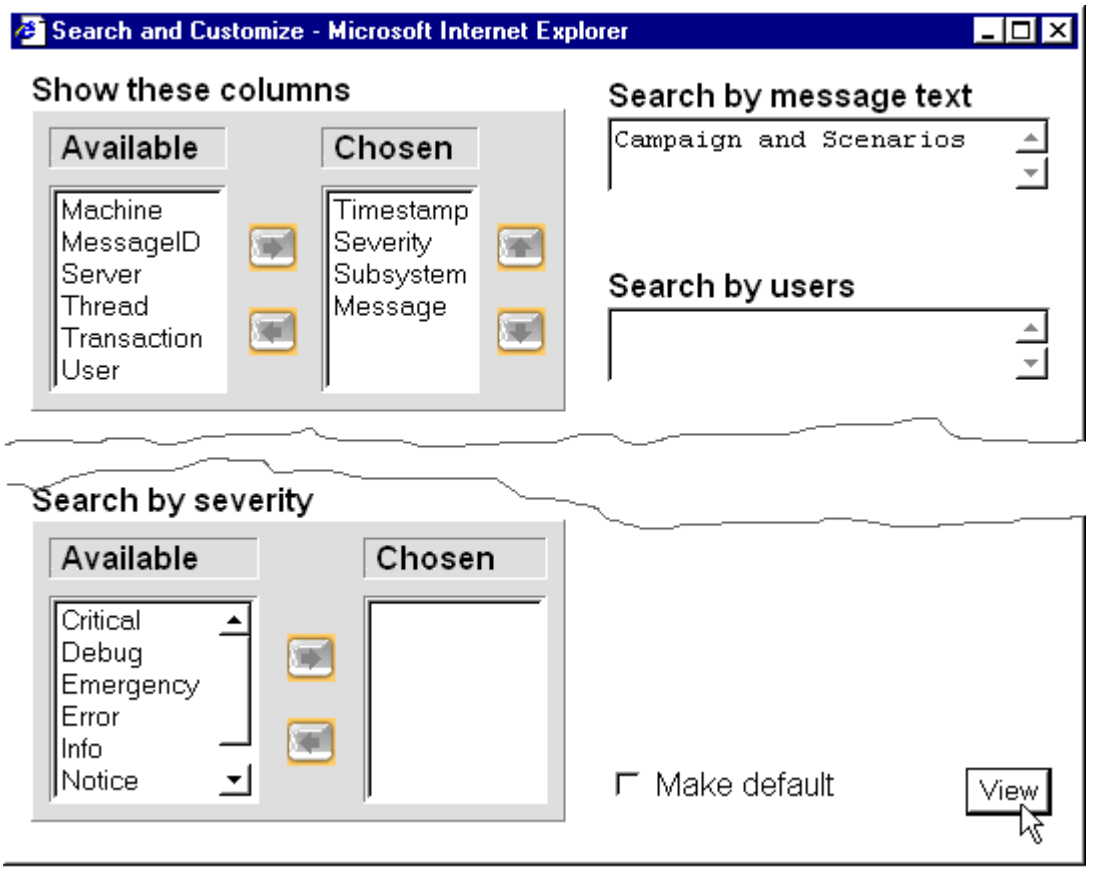
   The Administration Console displays another window titled Search and Customize.

6. In the Search and Customize window, in the Search By Message Text box, enter the following string:

   ```
   Campaign and Scenarios
   ```

7. Click View. (See Figure 1-6.)

Figure 1-6 The Search and Customize Window



The Administration Console closes the Search and Customize window and displays any campaign related messages in the Log window.

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
- Tip: Clearing Scenario Queries from an Ad Placeholder

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. Campaign Manager for WebLogic 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 2-7.

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 2-7.

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.

Table 2-2 Attributes for Image Files (Continued)

Attribute Name	Value Type	Description and Recommendations
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`.
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`.
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`.

Table 2-3 Attributes for Shockwave Files (Continued)

Attribute Name	Value Type	Description and Recommendations
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 Campaign Manager for WebLogic, refer to "Creating and Managing Content" in the *Guide to Building Personalized Applications*.

Loading Ads into the Reference Content Management System

Campaign Manager for WebLogic 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 Campaign Manager for WebLogic, 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 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:

WL_COMMERCE_HOME/dmsBase/Ads

where WL_COMMERCE_HOME is the directory in which you installed Campaign Manager for WebLogic.

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:

`WL_COMMERCE_HOME\bin\win32\loadads.bat` (Windows)
`WL_COMMERCE_HOME/bin/`*`UNIX-platform-type`*`/loadads.sh` (UNIX)

where `WL_COMMERCE_HOME` is the directory in which you installed Campaign Manager for WebLogic.

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 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:

- Add the New Type to the Deployment Descriptor
- Create and Compile a Java Class to Generate HTML
- Register the New Class in weblogiccommerce.properties

Add the New Type to the Deployment Descriptor

Each Campaign Manager for WebLogic Web application must specify its deployment requirements in an XML file called a deployment descriptor. To add a new MIME type for ad placeholders, you must modify the deployment descriptor for your WebLogic Personalization Server Web application. You can use a text editor to modify the deployment descriptor.

If you use the example portal as a framework for developing your own Web application, then the deployment descriptor is located at the following pathname:

```
WL_COMMERCE_HOME/config/wlcsDomain/applications/wlcsApp/exampleportal/WEB-INF/web.xml
```

where `WL_COMMERCE_HOME` is the location in which you installed Campaign Manager for WebLogic. Your Web application might be in another location. Contact your Campaign Manager for WebLogic administrator for information on which deployment descriptor to modify.

The deployment descriptor for your WebLogic Personalization Server Web application already contains a set of mappings for MIME type. Before you add a new type, review the existing mappings. Listing 2-3 illustrates a single MIME mapping from the example portal's deployment descriptor.

Listing 2-3 MIME Mapping in exampleportal/WEB-INF/web.xml

```
<mime-mapping>

    <extension>
      jpeg
    </extension>

    <mime-type>
      image/jpeg
    </mime-type>

</mime-mapping>
```

To add a new mapping, use the following syntax:

```
<mime-mapping>

    <extension>
      file-extension
    </extension>

    <mime-type>
      type/subtype
    </mime-type>

</mime-mapping>
```

where *file-extension* is the extension of the file type you want to map and *type/subtype* is a recognized MIME type and subtype.

Make sure that you provide end-tags for each of the XML elements.

When you save the modified deployment descriptor, you must restart the server to deploy the modifications. However, we recommend that you do not restart the server until you have registered the new Java class in `weblogiccommerce.properties` as described in "Register the New Class in weblogiccommerce.properties" on page 2-14.

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 `bea/commerce/platform/ad/AdContentProvider` interface. For information on the `bea/commerce/platform/ad/AdContentProvider` interface, refer to Campaign Manager for WebLogic Javadoc.

After you compile the class, you must save it in or below a directory that is specified in the system's `CLASSPATH` environment variable. For example `WL_COMMERCE_HOME/classes` is in the classpath. 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 in weblogiccommerce.properties

After you save the class in a directory that is in your classpath, you must notify Campaign Manager for WebLogic of its existence and purpose by adding a line to `weblogiccommerce.properties`. You can use a text editor to modify this file, which is located at the following pathname:

`WL_COMMERCE_HOME/weblogiccommerce.properties`

where `WL_COMMERCE_HOME` is the location in which you installed Campaign Manager for WebLogic.

To register your new class in the `weblogiccommerce.properties` file, find the section that Listing 2-4 illustrates. Then add a line that conforms to the following syntax:

`adtargettag.rendering.`*`mime-type`*`.`*`mime-extension`*`=`*`your-classname`*

Provide the following values for the variables in the previous syntax statement:

- *`mime-type`*. The name of the MIME type that you want to support.
- *`mime-extension`*. The filename extension that Campaign Manager for WebLogic uses to associate the file with the MIME type.
- *`your-classname`*. 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.

For example, `WL_COMMERCE_HOME/classes` is in the classpath. You saved your class to support AVI files as `WL_COMMERCE_HOME/classes/myclasses/MimeAvi.class` To register your classname, add the following line to `weblogiccommerce.properties`:

```
adtargettag.rendering.video.avi=myclasses.MimeAvi
```

Listing 2-4 Rendering Classes in weblogiccommerce.properties

```
################################################
# AdTargetTag Properties

adtargettag.rendering=com.bea.commerce.platform.ad.AdClickThruSer
vlet
# This is the class that implements the AdEventTracker interface
# and is used to raise events
adtargettag.eventtracking=com.bea.commerce.campaign.AdTracking

# Additional classes to render content based upon mime type
# To use replace the "text.html" with the mime type, replacing any
#   '/' characters with '.'
# Place the name of the java class that handles the mime type after
the '='

#adtargettag.rendering.text.html=
```

Tip: Clearing Scenario Queries from an Ad Placeholder

In the E-Business Control Center, a BA can create a scenario action that places a query in a specific ad placeholder when an event or some other state triggers the scenario.

Campaign Manager for WebLogic uses a table to keep track of which scenario-specific queries in a placeholder apply to which customer IDs. For example, if Pat Gomes triggers a scenario to place a query in the top-banner placeholder, Campaign Manager for WebLogic stores the Pat Gomes/top-banner-placeholder/scenario-query association in a table in its data repository.

Once a scenario places a query in a placeholder for a specific user, Campaign Manager for WebLogic persists the information until any of the following occurs:

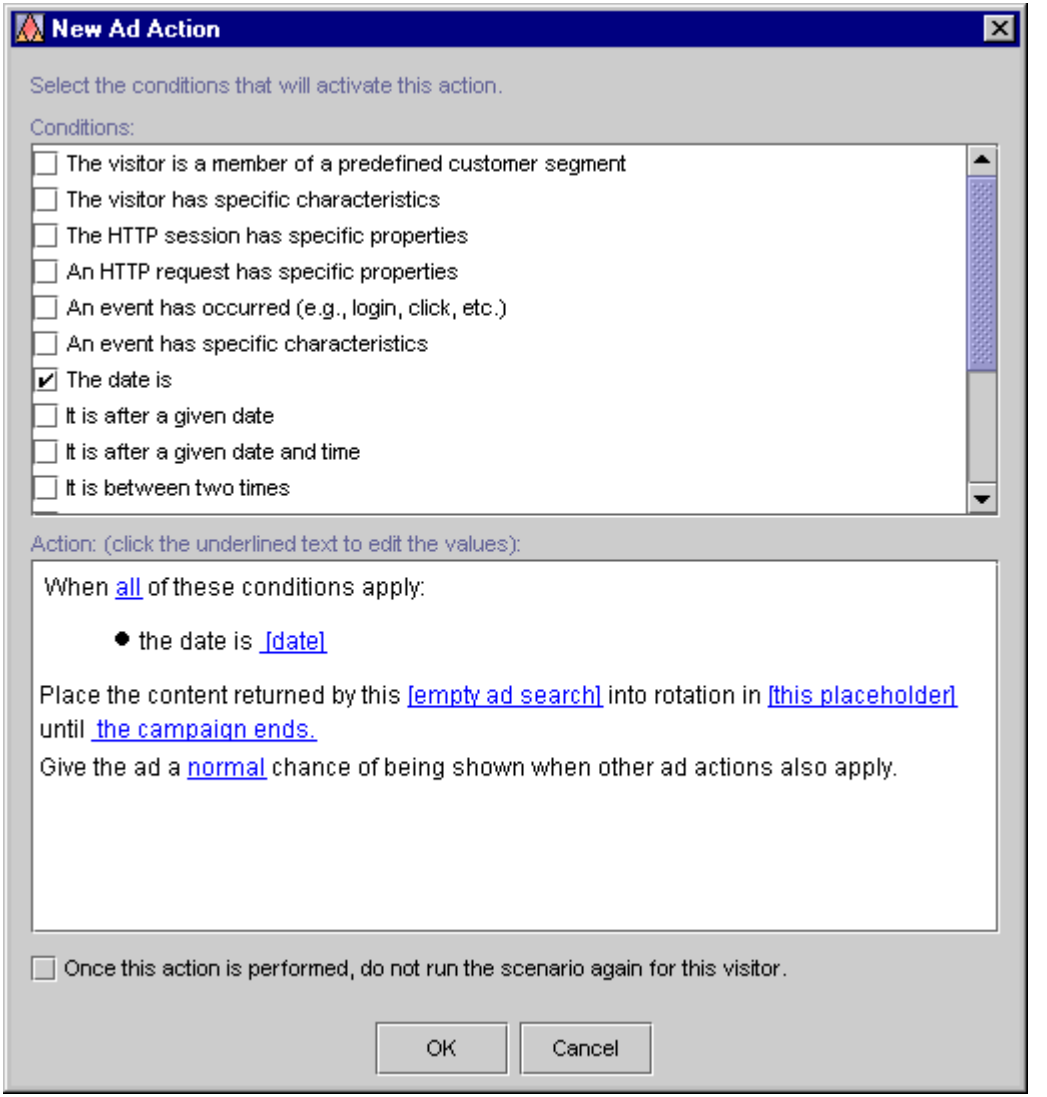
- The campaign achieves its goal for running the scenario query and displaying the corresponding ad.
- The campaign ends because it has achieved some other goal, the end date/time for the campaign has arrived, or a BA ends the campaign in the E-Business Control Center.
- An event or system state triggers another action for the scenario (or re-triggers the same action) that placed the query in the placeholder.

  Scenarios can contain multiple actions that place queries in one or more placeholders under different conditions. For example, when Pat Gomes logs in, scenario1 can place queryA in the top-banner placeholder. When Pat Gomes adds a drill to the shopping cart, scenario1 can place queryB in the top-banner placeholder. However, when a scenario engages a scenario action, it can place only one query in a placeholder for a given customer, and the query remains in the placeholder until the scenario replaces it. For example, when Pat Gomes adds a drill to the shopping cart, scenario1 replaces its queryA with queryB.
- The date/time that you specify in the Select Duration window in the E-Business Control Center has expired.

  The Select Duration window is available for any queries that you create after you install Service Pack 1 or later of Campaign Manager for WebLogic, WebLogic Commerce Server, and WebLogic Personalization Server. You access

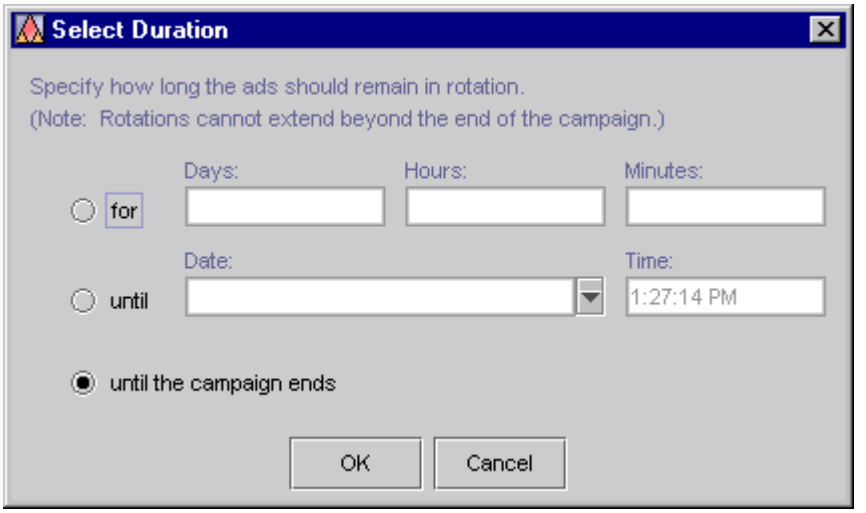
the window as part of creating a query for an ad placeholder. From the New Ad Action window, under Action, click the “until the campaign ends” link. (See Figure 2-1.)

Figure 2-1 Accessing the Select Duration Window



In the Select Duration window, indicate whether you want the query to stop running before the campaign ends by selecting a timeframe or date. (See Figure 2-2.)

Figure 2-2 The Select Duration Window



Because of the persistence mechanism, some actions might leave a scenario query stranded in a placeholder for a specific user, even across multiple sessions on the Web site.

For example:

- After a scenario action places a query in the top-banner placeholder for Pat Gomes, a BA deletes the scenario action. Deleting the scenario action does not clear the query from the placeholder. Only another action from the same scenario can replace the query. Instead of deleting the scenario or scenario action, we recommend that you modify the scenario action to place default queries in the placeholder. Then, when the scenario action is triggered, it replaces the campaign-specific query with a generic, default ad query. (A BA defines default queries in the Ad Placeholder Editor, which is part of the E-Business Control Center.)
- Pat Gomes places a drill in the shopping cart and triggers a scenario action to place a query in a shopping-cart placeholder. If Pat Gomes removes the drill from the shopping cart, the scenario action does not remove its query from the shopping-cart placeholder. Only another action from the same scenario can replace the query. For example, along with the action that places a query in the shopping-cart placeholder when a user adds a drill to the shopping cart, you can create an action for the same scenario that places default queries in the shopping-cart placeholder when a user removes a drill from the shopping cart.

In general, we recommend that for each scenario that includes an ad-placeholder action, you include an additional scenario action that does the following:

- When a user logs in, display default queries in the same ad placeholder.

This scenario action clears the placeholder from any campaign-specific ad queries each time a customer logs on to the site.

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 Campaign Manager for WebLogic services.

This topic includes the following sections:

- Creating User Profile Tags to Support Events
- 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.”

Creating User Profile Tags to Support Events

To provide information about the customer who interacts with the site, the Event Service must use the `<um:getProfile>` JSP tag. This tag retrieves information from the customer profile, making it available to the JSP on which the tag resides.

To Create a Get Profile 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="um.tld" prefix="um" %>
   ```

3. Locate the get profile tag somewhere below the statement to import the `um.tld` tag library. If any other JSP tag requires the information in the customer profile, locate the get profile tag before the other JSP tag.
4. Use the following syntax to create the get profile tag:

   ```
   <um:getProfile profileKey="name"/>
   ```

   where *name* provides a unique identifier that the Event Service can use to retrieve the profile. If other tags on the JSP require information in the customer profile, you must use additional attributes in the tag. For more information, refer to `<um:getProfile>` under "JSP Tag Library Reference" in the *Guide to Building Personalized Applications*.

Using Ad Placeholder Tags to Display Ads

After a BA uses the E-Business Control Center to create ad placeholders, a CBE 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:

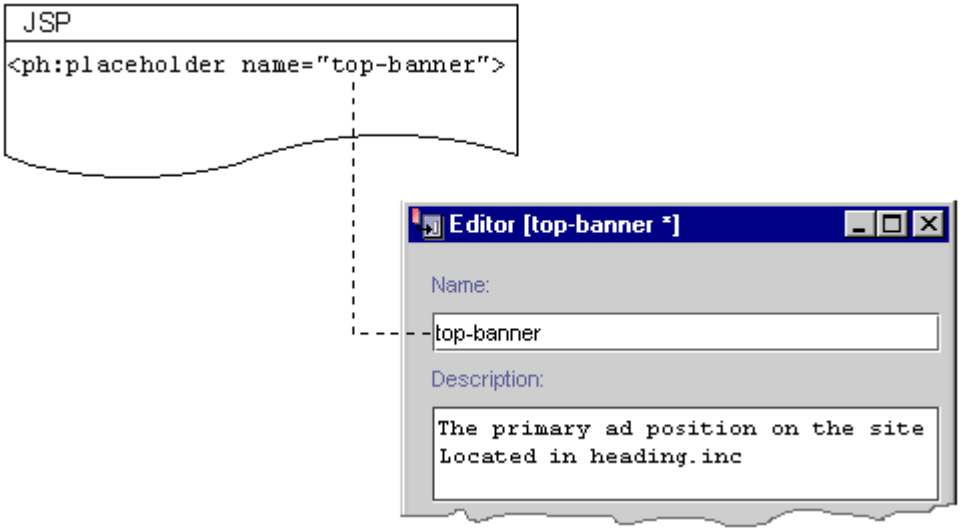
   ```
   <%@ taglib uri="ph.tld" prefix="ph" %>
   ```

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



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

All JSP files in the e-commerce sample web application include `heading.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 Campaign Manager for WebLogic renders the placeholder in the `main.jsp` file, which is the home page for the e-commerce JSP templates.

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 2-11.
- Use the document attributes that are described in "Specifying Display and Clickthrough Behavior" on page 2-2.
- 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 Developing 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

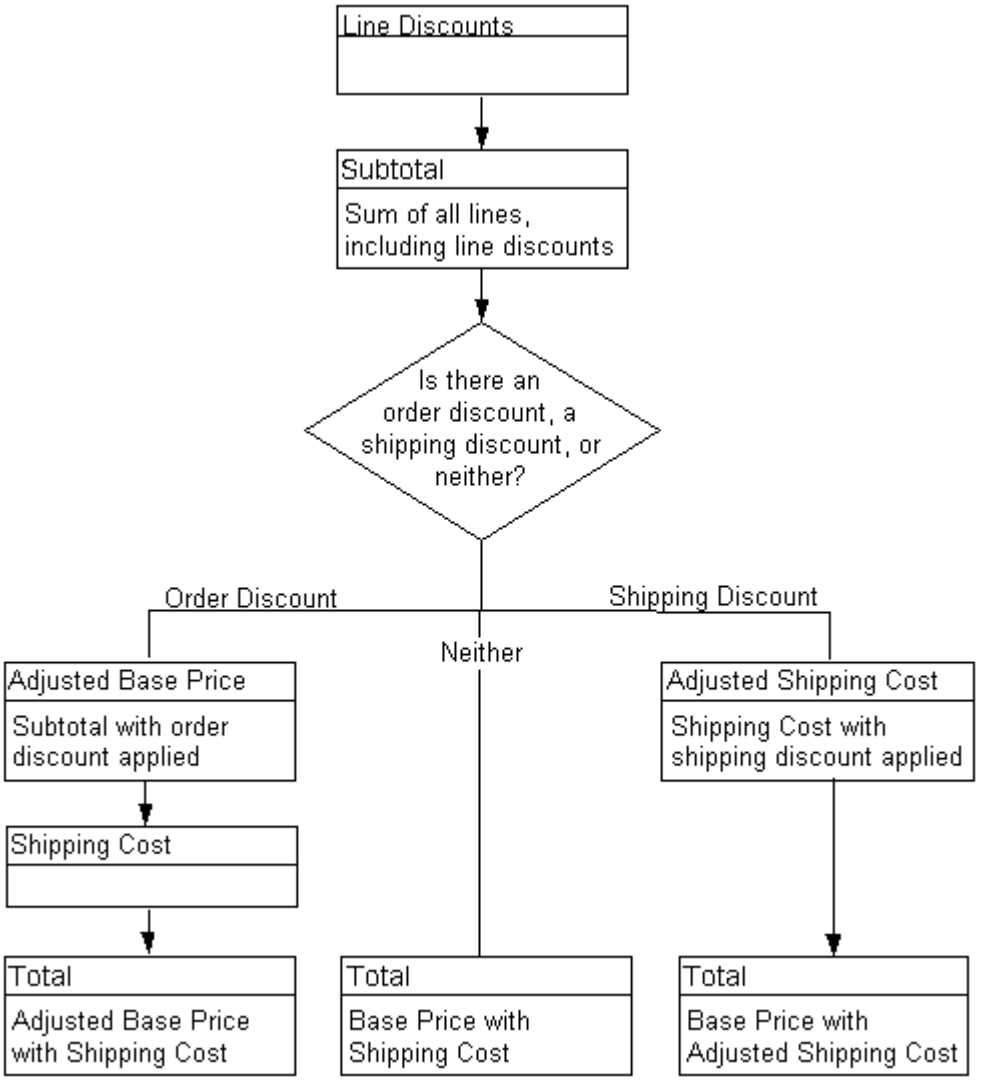
The Pricing Service follows this sequence for applying discounts (see Figure 3-3):

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



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 Campaign Manager for WebLogic 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

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.)

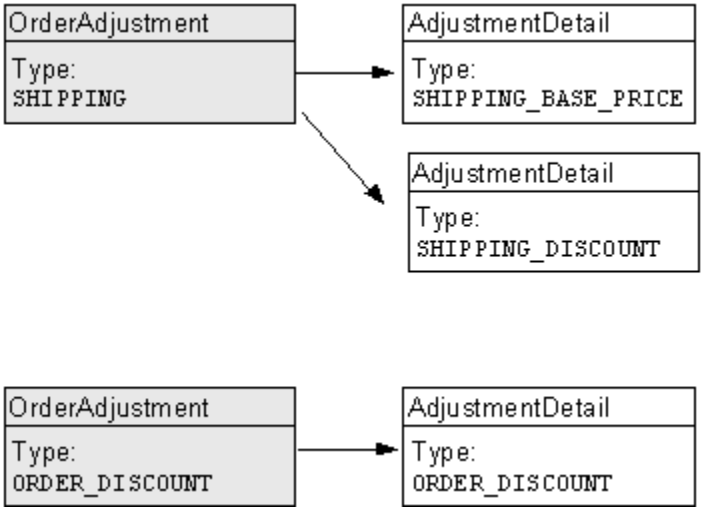
Note that a shopping cart may have an order discount or a shipping discount, but not both. For any given shopping cart with order or shipping discounts, the Price Service creates **one** of the following:

- 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 3-12.

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 CBEs.

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 Campaign Manager for WebLogic 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 CBEs.

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`.

Table 3-3 AdjustmentDetail Methods (Continued)

Method	Returns Data of Type	Description
`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 Campaign Manager for WebLogic 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"

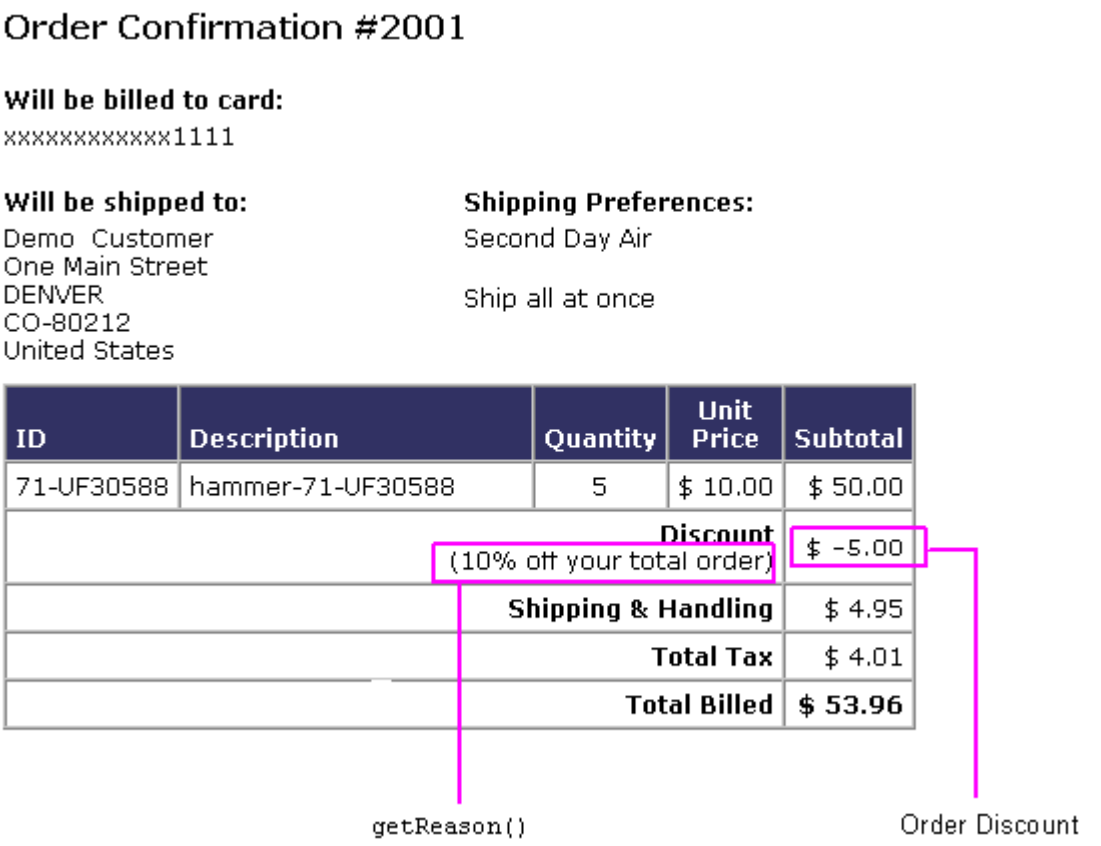
The Pricing Service also creates an `AdjustmentDetail` object to supplement the information in the `OrderAdjustment` object. The `AdjustmentDetail` object contains the following information:

- `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



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:

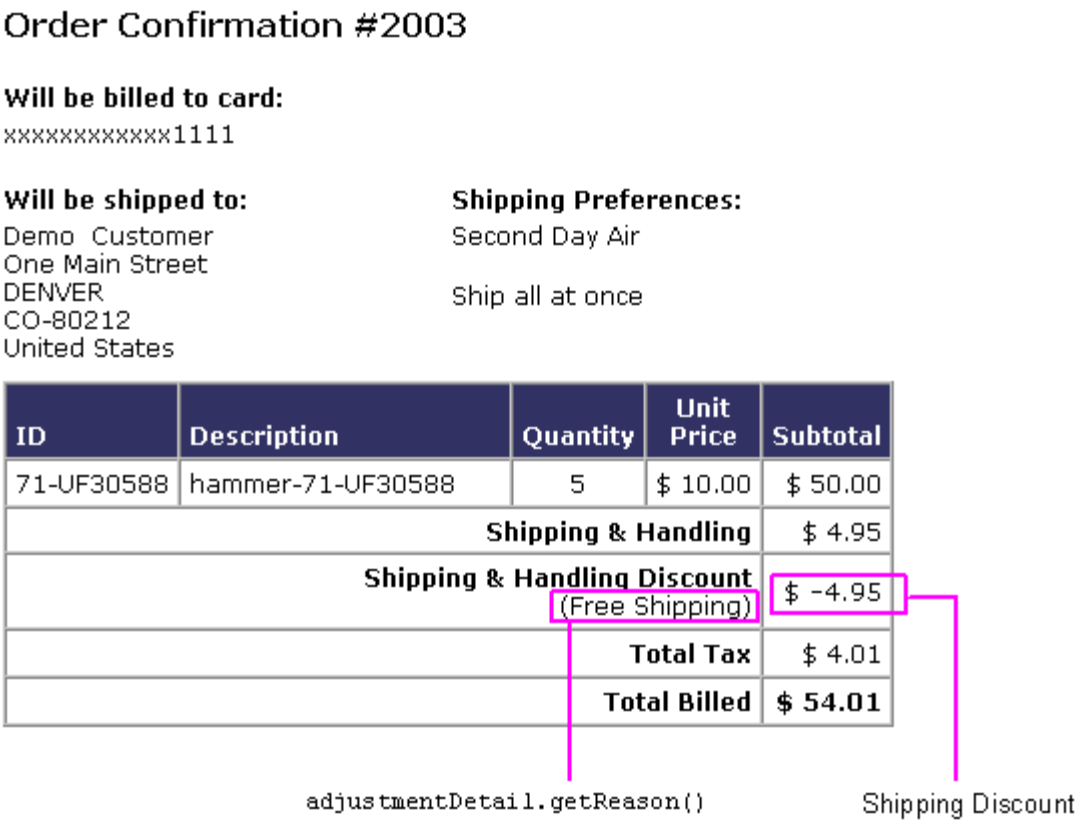
- `getType()`: SHIPPING_BASE_PRICE
- `getInitialPrice()`: $50
- `getEndPrice()`: $54.95
- `getComputation()`: ""
- `getReason()`: "Base Shipping Charges"
- `getCurrency()`: "USD"

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



4 Setting Up and Sending E-mail for Campaigns

Campaign Manager for WebLogic 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
- Setting Properties for the Mail Service
- Creating E-mail JSPs
- Sending Bulk Mail

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

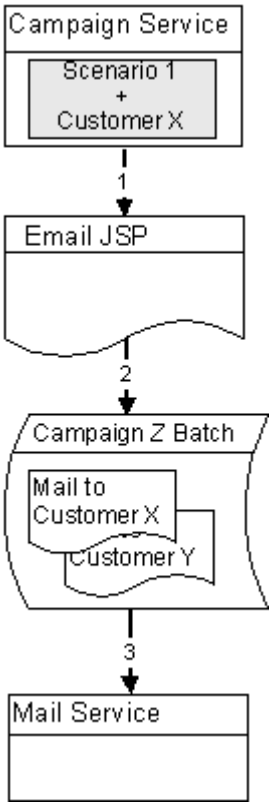
How Campaigns Use the Mail Service

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 provide the URI of 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 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 Campaign Manager for WebLogic 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



Setting Properties for the Mail Service

Before a campaign can send e-mail, you must set properties for the Mail Service. In a clustered domain, you must set these properties on each machine in the cluster.

To set properties, do the following:

1. Open the Administration Tool and find the following:
   - 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. By default, this information is stored 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*.
2. From a text editor, open the following file:

   `WL_COMMERCE_HOME/weblogiccommerce.properties`

   where `WL_COMMERCE_HOME` is the directory in which you installed Campaign Manager for WebLogic.
3. In `weblogiccommerce.properties`, find the section that begins with the following line (see Listing 4-1):

   ```
   # MailService and MailAction properties
   ```

4. Enter values for the following properties:
   - `mail.smtpHost`. Enter the name of the SMTP host (the server that sends e-mail).
   - `mailAction.defaultFromAddress`. Enter 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. A scenario-provided From address overrides this `mailAction.defaultFromAddress` property.
   - `mailAction.emailPropertySet`. Enter the name of the property set that contains customer e-mail properties.
   - `mailAction.emailPropertyName`. Enter the name of the property that contains customer e-mail addresses.
5. By default, the Mail Service directs requests to `localhost`. However a clustered environment uses a single proxy server to accept HTTP requests (including HTTP requests from the Mail Service). In a clustered environment, you must specify the name and port number of the proxy server by removing the comment characters from the following properties and entering values that are appropriate for your environment:
   - `mailAction.jspHost`. Enter the name of the cluster's proxy server.
   - `mailAction.jspPort`. Enter the port to use when connecting to the host that you specified in `mailAction.jspHost`.
6. If you use a different property set and property to record customer's campaign-related e-mail preferences, modify the values of `mailAction.optInPropertySet` and `mailAction.optInPropertyName`.

Listing 4-1 Mail Properties in weblogiccommerce.properties

```
##############################################

# MailService and MailAction properties

# SMTP Host for the MailService to use
mail.smtpHost=walawala.sprockets.com

# A default from-address, if none is provided to the MailAction
mailAction.defaultFromAddress=acme@sprockets.com

# The property set to query for the user's email address
mailAction.emailPropertySet=Customer Properties

# The name of the property to query for the user's email address
mailAction.emailPropertyName=email

# The property set that contains the email opt-in property
mailAction.optInPropertySet=Demographics

# The name of the email opt-in property. If a user sets this to
# false, MailAction will not send any emails to that user.
mailAction.optInPropertyName=Email_Opt_In

# The name of the server that contains the email-generating JSP's
# (overriding this should only be necessary in a cluster)
#mailAction.jspHost=localhost

# The port of the server that contains the email-generating JSP's
# (overriding this should only be necessary in a cluster)
#mailAction.jspPort=7501
```

7. Save your modifications to `weblogiccommerce.properties`.
8. To use the modified properties, restart the server. The server only refers to its properties file during the startup process.

Creating E-mail JSPs

The E-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 Campaign Manager for WebLogic.

You must save the JSP on your Campaign Manager for WebLogic host in a location that Campaign Manager for WebLogic processes can access.

This section describes the following:

- E-mail Parameters
- Sample E-mail JSP

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.

Sample E-mail JSP

Listing 4-2 shows an example of an e-mail JSP, which does the following:

- Imports the user management tag library (`um.tld`).
- Uses `request.getParameter` object to retrieve the contents of the `userId` parameter.
- Passes the `userId` to the `<um:getProfile>` JSP tag to get the user's profile.
- Uses the `<um:getProperty>` to display the user's first and last name, and the user's home city and state.

Listing 4-2 Sample E-mail JSP

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

<%
  String userId = (String)request.getParameter( "userId" );
  if ( userId == null )
  {
    throw new ServletException( "A userId is required to generate this email" );
  }
%>

<um:getProfile profileKey="<%= userId %>" />

Dear
<um:getProperty propertyName="FirstName"/>
<um:getProperty propertyName="LastName"/>,

Thank you for shopping at our store! We noticed that you are from

<um:getProperty propertyName="contactAddressCity"/>,
<um:getProperty propertyName="contactAddressState"/>,

and thought you would be interested in purchasing tickets to the Yankees-Red Sox
game...
```

Sending Bulk Mail

You must periodically use a command to send the batches e-mail that the JSPs store in the Campaign Manager for WebLogic 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 Campaign Manager for WebLogic 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 Campaign Manager for WebLogic 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 Campaign Manager for WebLogic host), do the following:

1. Open the following file in a text editor:
   `WL_COMMERCE_HOME\bin\win32\mailmanager.bat` (Windows)
   `WL_COMMERCE_HOME/bin/win32/mailmanager.sh` (UNIX)
2. In the `mailmanager` script, in the `SET HOST=` line, replace `localhost` with the name of a Campaign Manager for WebLogic 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:
   `WL_COMMERCE_HOME\bin\win32\mailmanager.bat` (Windows)
   `WL_COMMERCE_HOME/bin/win32/mailmanager.sh` (UNIX)
2. In the `mailmanager` script, in the `SET HOST=` line, replace `localhost` with the name of a Campaign Manager for WebLogic 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 Campaign Manager for WebLogic host:

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

   `MailManager.bat list` (Windows)

   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 send-delete` *`batch-name`*

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 list`

   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 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 4-6.

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.commerce.platform.mail.MailManager`.

The command syntax is as follows:

`MailManager.bat [ list | send | send-delete | delete ] [ batch-name ]` (Windows)

`MailManager.sh [ list | send | send-delete | delete ] [ batch-name ]` (UNIX)

If you do not specify command arguments, MailManager prints to standard output the names all e-mail batches in the data repository and the number of e-mails in each batch.

Use the command arguments as follows:

- `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 `campaign1`, enter the following command:
`MailManager.bat list campaign1`

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

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

CHAPTER

5 Campaign Manager Database Schema

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

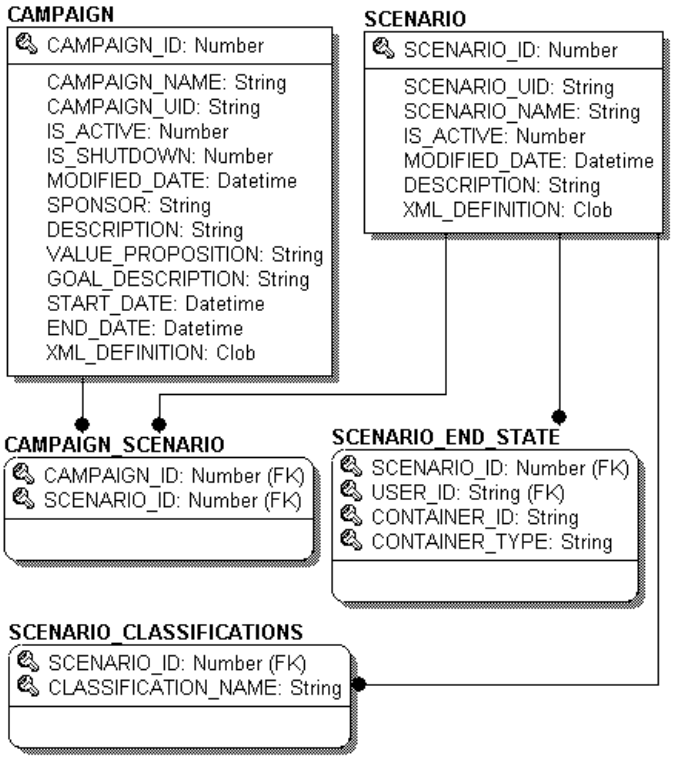
This topic includes the following sections:

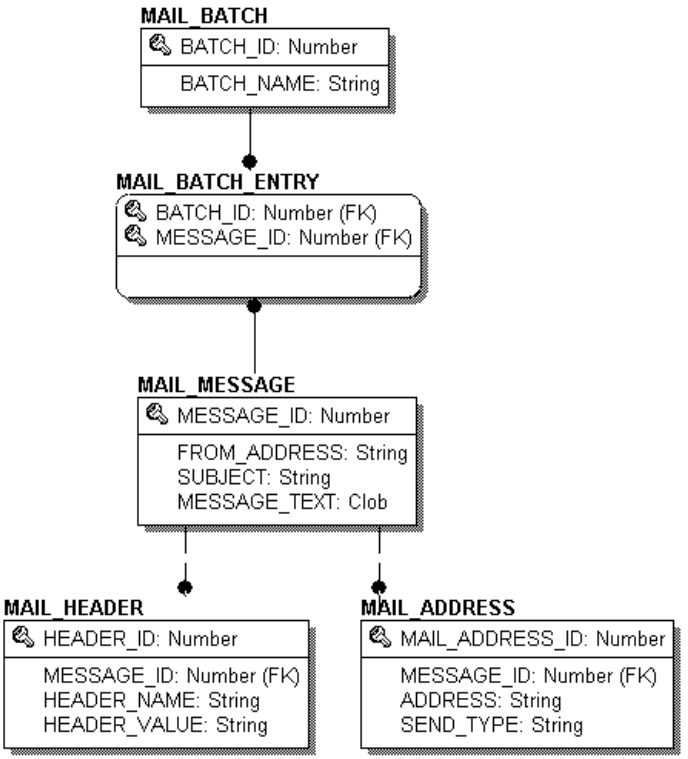
- The Entity-Relation Diagram
- List of Tables Comprising the BEA Campaign Manager
- The Campaign Manager Data Dictionary
- The SQL Scripts Used to Create the Database
- Defined Constraints

The Entity-Relation Diagram

Table 5-1 shows the Entity-Relation diagram for the E-Business Control Center for WebLogic campaign manager 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




MAIL_BATCH
BATCH_ID: Number
BATCH_NAME: String
MAIL_BATCH_ENTRY
BATCH_ID: Number (FK)
MESSAGE_ID: Number (FK)
MAIL_MESSAGE
MESSAGE_ID: Number
FROM_ADDRESS: String
SUBJECT: String
MESSAGE_TEXT: Clob
MAIL_HEADER
HEADER_ID: Number
MESSAGE_ID: Number (FK)
HEADER_NAME: String
HEADER_VALUE: String
MAIL_ADDRESS
MAIL_ADDRESS_ID: Number
MESSAGE_ID: Number (FK)
ADDRESS: String
SEND_TYPE: String

List of Tables Comprising the BEA Campaign Manager

The BEA Campaign Manager is comprised of the following tables. In this list, the tables are sorted by functionality:

Campaign and Scenarios

- The CAMPAIGN Database Table
- The CAMPAIGN_SCENARIO Database Table
- The SCENARIO Database Table
- The SCENARIO_CLASSIFICATIONS Database Table
- The SCENARIO_END_STATE Database Table

Mail

- The MAIL_ADDRESS Database Table
- The MAIL_BATCH Database Table
- The MAIL_BATCH_ENTRY Database Table
- The MAIL_HEADER Database Table
- The MAIL_MESSAGE Database Table

The Campaign Manager Data Dictionary

In this section, the Campaign Manager schema tables are arranged alphabetically as a data dictionary.

Note: Even though the following documentation references "foreign keys" to various tables, these constraints do not currently exist in this release of Campaign Manager for WebLogic. However, they will be (available in future releases) in place in future versions of Campaign Manager for WebLogic and we want you to be aware of these relationships now.

The CAMPAIGN Database Table

Table 5-1 describes the metadata for the E-Business Control Center CAMPAIGN table. This table stores all of the campaign related information.

The Primary Key is `CAMPAIGN_ID`.

Table 5-1 CAMPAIGN Table Metadata

Column Name	Data Type	Description and Recommendations
`CAMPAIGN_ID`	`NUMBER(15)`	PK—a unique, system-generated number to be used as the record ID.
`CAMPAIGN_NAME`	`VARCHAR(50)`	The name of the campaign.
`CAMPAIGN_UID`	`VARCHAR(50)`	A textual unique identifier for the campaign record.
`IS_ACTIVE`	`NUMBER(1)`	Active=1, Non-active=0
`IS_SHUTDOWN`	`NUMBER(1)`	Shutdown=1, Not shutdown=0
`MODIFIED_DATE`	`DATE`	The date the campaign record was last modified.
`SPONSOR`	`VARCHAR(50)`	The sponsor of the campaign.
`DESCRIPTION`	`VARCHAR(254)`	The description of the campaign.

Table 5-1 CAMPAIGN Table Metadata (Continued)

Column Name	Data Type	Description and Recommendations
VALUE_PROPOSITION	VARCHAR(254)	The value proposition of the campaign.
GOAL_DESCRIPTION	VARCHAR(254)	The intended goal of the campaign.
START_DATE	DATE	The starting date and time of the campaign.
END_DATE	DATE	The ending date and time of the campaign.
XML_DEFINITION	CLOB	This is XML.

The CAMPAIGN_SCENARIO Database Table

Table 5-2 describes the metadata for the E-Business Control Center CAMPAIGN_SCENARIO table. This table is a cross-reference table to identify which scenarios are associated with which campaigns.

CAMPAIGN_ID and SCENARIO_ID together form the primary key for the CAMPAIGN_SCENARIO table.

Table 5-2 CAMPAIGN_SCENARIO Table Metadata

Column Name	Data Type	Description and Recommendations
CAMPAIGN_ID	NUMBER(15)	The primary key of the campaign record.
SCENARIO_ID	NUMBER(15)	The primary key of the scenario record.

The MAIL_ADDRESS Database Table

Table 5-3 describes the metadata for the E-Business Control Center MAIL_ADDRESS table. This table stores all of the address info for e-mail purposes.

The Primary Key is MAIL_ADDRESS_ID.

Table 5-3 MAIL_ADDRESS Table Metadata

Column Name	Data Type	Description and Recommendations
MAIL_ADDRESS_ID	NUMBER(15)	PK—a unique, system-generated number to be used as the record ID.
MESSAGE_ID	NUMBER(15)	FK—foreign key to the MAIL_MESSAGE table.
ADDRESS	VARCHAR(254)	Stores the various e-mail addresses on the distribution list.
SEND_TYPE	VARCHAR(4)	Determines how the ADDRESS should be included on the distribution. Possible values are TO, CC, or BCC.

The MAIL_BATCH Database Table

Table 5-4 describes the metadata for the E-Business Control Center MAIL_BATCH table. This table establishes a batch for each mailing.

The Primary Key is BATCH_ID.

Table 5-4 MAIL_BATCH Table Metadata

Column Name	Data Type	Description and Recommendations
BATCH_ID	NUMBER(15)	PK—a unique, system-generated number to be used as the record ID.
BATCH_NAME	VARCHAR(50)	The name of the mail message batch.

The MAIL_BATCH_ENTRY Database Table

Table 5-5 describes the metadata for the E-Business Control Center MAIL_BATCH_ENTRY table. This table is used to correlate the mail batch with the specific mail message.

The Primary Keys are BATCH_ID and MESSAGE_ID.

Table 5-5 MAIL_BATCH_ENTRY Table Metadata

Column Name	Data Type	Description and Recommendations
BATCH_ID	NUMBER(15)	PK and FK—a unique, system-generated number to be used as the record ID.
MESSAGE_ID	NUMBER(15)	PK and FK—foreign key to the MAIL_MESSAGE table.

The MAIL_HEADER Database Table

Table 5-6 describes the metadata for the E-Business Control Center MAIL_HEADER table. This table contains all of the header information specific to the e-mail message.

The Primary Key is HEADER_ID.

Table 5-6 MAIL_HEADER Table Metadata

Column Name	Data Type	Description and Recommendations
HEADER_ID	NUMBER(15)	PK—a unique, system-generated number to be used as the record ID.
MESSAGE_ID	NUMBER(15)	FK—foreign key to the MAIL_MESSAGE table.
HEADER_NAME	VARCHAR(50)	The name of the mail message header.
HEADER_VALUE	VARCHAR(254)	The value of the mail message header.

The MAIL_MESSAGE Database Table

Table 5-7 describes the metadata for the E-Business Control Center MAIL_MESSAGE table. This table contains the specifics of the mail message (e.g., the subject line, text, etc.).

The Primary Key is MESSAGE_ID.

Table 5-7 MAIL_MESSAGE Table Metadata

Column Name	Data Type	Description and Recommendations
MESSAGE_ID	NUMBER(15)	PK—a unique, system-generated number to be used as the record ID.
FROM_ADDRESS	VARCHAR(254)	Identifies who is sending the message.
SUBJECT	VARCHAR(128)	Stores the mail message subject.
MESSAGE_TEXT	CLOB	Holds the content of the mail message.

The SCENARIO Database Table

Table 5-8 describes the metadata for the E-Business Control Center SCENARIO table. This table is used to store specific scenario information.

The Primary Key is SCENARIO_ID.

Table 5-8 SCENARIO Table Metadata

Column Name	Data Type	Description and Recommendations
SCENARIO_ID	NUMBER(15)	PK—a unique, system-generated number to be used as the record ID.
SCENARIO_UID	VARCHAR(50)	A textual unique identifier for the scenario record.
SCENARIO_NAME	VARCHAR(50)	The name of the scenario.
IS_ACTIVE	NUMBER(1)	Active=1, Non-active=0
MODIFIED_DATE	DATE	The date the scenario record was last modified.
DESCRIPTION	VARCHAR(254)	The description of the scenario.
XML_DEFINITION	CLOB	This is XML.

The SCENARIO_CLASSIFICATIONS Database Table

Table 5-9 describes the metadata for the E-Business Control Center SCENARIO_CLASSIFICATIONS table. This table is used to store customer segment info for each scenario.

The Primary Keys are SCENARIO_ID and CLASSIFICATION_NAME.

Table 5-9 SCENARIO_CLASSIFICATIONS Table Metadata

Column Name	Data Type	Description and Recommendations
SCENARIO_ID	NUMBER(15)	FK—the unique identifier of a scenario.
CLASSIFICATION_NAME	VARCHAR(100)	The customer segment name for the scenario to which it is related.

The SCENARIO_END_STATE Database Table

Table 5-10 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_ID, USER_ID, CONTAINER_ID, and CONTAINER_TYPE.

Table 5-10 SCENARIO_END_STATE Table Metadata

Column Name	Data Type	Description and Recommendations
SCENARIO_ID	NUMBER(15)	FK—the scenario unique identifier.(SCENARIO.SCENARIO_UID)
USER_ID	VARCHAR(50)	FK—the user ID. (WLCS_SUER.IDENTIFIER)
CONTAINER_ID	VARCHAR(50)	FK—the campaign unique identifier. (CAMPAIGN.CAMPAIGN_UID)
CONTAINER_TYPE	VARCHAR(50)	At this time this column will always hold the string 'Campaign'.

The SQL Scripts Used to Create the Database

The database schemas for the WebLogic Personalization Server, WebLogic Commerce Server and BEA's Campaign Manager for WebLogic are all created by executing the `create_all` script for the target database environment.

Cloudscape

For Cloudscape, execute one of the following:

- `WL_COMMERCE_HOME\db\cloudscape\3.5.1\create_all.bat` (Windows)
- `WL_COMMERCE_HOME/db/cloudscape/3.5.1/create_all.sh` (UNIX)

Script Name	Description
`create_all.bat`	The execution of this script will create the WLPS, WLCS and Campaign Manager database schema.
`create_all.sh`	The execution of this script will create the WLPS, WLCS and Campaign Manager database schema.
`create_campaign.sql`	Creates the Campaign Manager specific database objects (e.g., tables, indexes and constraints).
`create_common.sql`	Creates the database objects which are common to WLPS and WLCS.
`create_mail_ad.sql`	Creates all the database objects used by the mail messaging component.
`create_wlcs.sql`	Creates all the database objects for WLCS (including Catalog and Order Management).
`create_wlps.sql`	Creates all the database object for WLPS.
`drop_campaign.sql`	Drops all database objects associated with Campaign Manager.
`drop_common.sql`	Drops the database objects which are common between WLPS and WLCS.

Script Name	Description
drop_mail_ad.sql	Drops the database objects used by the mail messaging component.
drop_wlcs.sql	Drops the database objects associated with WLCS.
drop_wlps.sql	Drops the database objects associated with WLPS.
insert_common.sql	Inserts core data into the common tables between WLPS and WLCS.
insert_wlcs.sql	Inserts core data into some of the WLCS tables.
insert_wlcs_sample_catalog.sql	Inserts sample data into the product catalog.
insert_wlcs_sample_customer.sql	Inserts sample customer information into WLCS tables.
insert_wlcs_sample_data.sql	Inserts sample data into various WLCS tables.
insert_wlps.sql	Inserts core data into WLPS tables.
insert_wlps_sample_data.sql	Inserts sample data into various WLPS tables.

Oracle

For Oracle, from the command line, move to the following directory:

```
WL_COMMERCE_HOME/db/oracle/8.1.6
```

After logging into SQL*Plus, execute the `create_all.sql` script (e.g., `@create_all`).

Script Name	Description
create_campaign.sql	Creates the Campaign Manager specific database objects (e.g., tables, indexes and constraints).
create_common.sql	Creates the database objects which are common to WLPS and WLCS.

Script Name	Description
`create_mail_ad.sql`	Creates all the database objects used by the mail messaging component.
`create_wlcs.sql`	Creates all the database objects for WLCS (including Catalog and Order Management).
`create_wlps.sql`	Creates all the database object for WLPS.
`drop_campaign.sql`	Drops all database objects associated with Campaign Manager.
`drop_common.sql`	Drops the database objects which are common between WLPS and WLCS.
`drop_mail_ad.sql`	Drops the database objects used by the mail messaging component.
`drop_wlcs.sql`	Drops the database objects associated with WLCS.
`drop_wlps.sql`	Drops the database objects associated with WLPS.
`insert_common.sql`	Inserts core data into the common tables between WLPS and WLCS.
`insert_wlcs.sql`	Inserts core data into some of the WLCS tables.
`insert_wlcs_sample_catalog.sql`	Inserts sample data into the product catalog.
`insert_wlcs_sample_customer.sql`	Inserts sample customer information into WLCS tables.
`insert_wlcs_sample_data.sql`	Inserts sample data into various WLCS tables.
`insert_wlps.sql`	Inserts core data into WLPS tables.
`insert_wlps_sample_data.sql`	Inserts sample data into various WLPS tables.
`install_report.sql`	This script is used to summarize the database installation. Information such as the number of tables, indexes, etc., is displayed.
`statistics.sql`	This script is used in computing statistics on various database objects (e.g., tables and indexes) in an Oracle environment.

Defined Constraints

For some of the database tables described earlier in this chapter, the SQL files define constraints. Table 5-11 shows the table name and describes the constraint(s) defined for it.

Table 5-11 Constraints Defined on Campaign Manager Database Tables

Table Name	Constraints as Defined in create-catalog-oracle.sql
CAMPAIGN	A check constraint (CAMPAIGN_IS_ACTIVE) is used on the IS_ACTIVE column to ensure that the value of the column is either a 0 (false) or 1 (true). A check constraint (CAMPAIGN_IS_SHUTDOWN) is used on the IS_SHUTDOWN column to ensure that the value of the column is a 0 (false) or 1 (true).
CAMPAIGN_SCENARIO	A referential integrity constraint (FK1_CAMP_SCNR_CAMP) ensures that a CAMPAIGN record exists before the CAMPAIGN_SCENARIO record can be inserted. A referential integrity constraint (FK2_CAMP_SCNR_CAMP) ensures that a SCENARIO record exists before the CAMPAIGN_SCENARIO record can be inserted.
SCENARIO_CLASSIFICATIONS	A referential integrity constraint (FK1_SCNR_CLSS_SCNR) ensures that a SCENARIO record exists before the SCENARIO_CLASSIFICATIONS record can be inserted.
SCENARIO_END_STATE	A referential integrity constraint (FK1_SCNR_END_SCNR) ensures that a SCENARIO record exists before the SCENARIO_END_STATE record can be inserted. A referential integrity constraint (FK2_CAMP_SCNR_CAMP) ensures that a WLCS_USER record exists before the SCENARIO_END_STATE record can be inserted. A data integrity constraint exists so in the event that a user record is deleted from the WLCS_USER table then all records for that user will also be deleted from SCENARIO_END_STATE.

.