BEA WebLogic Commerce Server

Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline

BEA WebLogic Commerce Server 3.5
Document Edition 3.5
April 2001

Guide to Managing Presentation and Business Logic: Using Webflow and Pipeline

Document Edition	Date	Software Version
3.5	April 2001	BEA WebLogic Commerce Server 3.5

Contents

About This Document

This document provides information about the Webflow and Pipeline mechanisms included in the BEA WebLogic Commerce Server^{TM}. These mechanisms externalize the page flow and business logic that comprise any e-commerce Web site, and can be customized or extended to meet your business objectives.

This document includes the following topics:

- Chapter 1, "Overview of Webflow and Pipeline Management," which describes the high-level architecture and categories for the Webflow and Pipeline mechanisms utilized in the BEA WebLogic Commerce Server product.
- Chapter 2, "Customizing Webflow and Pipelines," which describes how a commerce engineer/JSP developer could customize the default Webflow and Pipeline mechanism to meet the requirements of their e-business.
- Chapter 3, "Using the Webflow and Pipeline Editor," which describes how a commerce engineer/JSP developer can use the Webflow and Pipeline Editor Administration Tool to edit and validate the `webflow.properties` and `pipeline.properties` files.
- Chapter 4, "Extending Webflow and Pipelines," which describes how a Java/EJB programmer can extend the default Webflow and Pipeline mechanisms to create new functionality for their e-business.
- Chapter 5, "Webflow and Pipeline JSP Tags Library Reference," which describes the specialized JSP tags that are used in the provided WebLogic Commerce Server Web application.

What You Need to Know

This document is intended for the following audiences:

- The commerce engineer/JSP content developer, who uses JSP templates and tag libraries to implement interactive Web pages to meet business requirements. This user also maintains simple configuration files.
- The business analyst, who defines the company's business protocols (processes and rules) for a business-to-consumer Web site. This user may set pricing policies and discounts, and may plan promotional advertising.
- The site administrator, who uses Commerce and Personalization Server administration screens to configure the site's rules, portals, property sets, user profiles, content delivery, and product catalog.
- The Java/EJB programmer, who creates custom code to insert in the JSP files. This user may also handle complex configuration files and write new Pipeline components or input processors.

e-docs Web Site

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

How to Print the Document

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

A PDF version of this document is available on the WebLogic Commerce 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 WebLogic Commerce 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

The following WebLogic Commerce Server documents describe parts of an e-commerce application built upon the Webflow and Pipeline infrastructure:

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

Contact Us!

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

In your e-mail message, please indicate that you are using the documentation for the WebLogic Commerce Server 3.5 release.

If you have any questions about this version of BEA WebLogic Commerce Server, or if you have problems installing and running BEA WebLogic Commerce Server, contact BEA Customer Support through BEA WebSUPPORT at **www.beasys.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. *Examples*: 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 Overview of Webflow and Pipeline Management

The Webflow and Pipeline are flexible mechanisms designed to help you manage both the presentation and business logic in your e-commerce Web site, without the need for advanced programming skills. This topic describes the high-level architecture of the Webflow and Pipeline, and provides preliminary information about how you can use these mechanisms to customize or extend the e-business site provided with BEA WebLogic Commerce Server.

This topic includes the following sections:

- High-level Architecture
- Development Roles
- Next Steps

High-level Architecture

The BEA WebLogic Commerce Server design model separates presentation (such as HTML and JavaScript) from business logic (such as database updates and implementation of business rules). To create and maintain this separation, the WebLogic Commerce Server makes use of the following six technologies:

- *HTML*: Standard HTML supported by Netscape Navigator or Microsoft Internet Explorer. Throughout this document, the term HTML refers to both HTML and JavaScript.
- *JSP Tags*: Customized tags used in the J2EE platform. The WebLogic Commerce Server uses JavaServer Page (JSP) tags to add dynamic display to the HTML pages, such as displaying the name of a customer who is currently logged in.
- *Pipeline Components*: Discrete units of server-side business logic, such as calculating tax or commiting an order. Pipeline components can be combined into a Pipeline.
- *Pipeline Session*: Storage location for information about the current session (such as the current shopping cart) or more transient data (such as error messages about a customer's most recent input).
- *Input Processors*: Flexible mechanisms that handle form submission. Some may perform validation of customer data, but the primary role of an input processor is to store customer data into the Pipeline session for subsequent use by a Pipeline component.
- *Webflow*: Controls the flow of a customer's session through the pages displayed in a browser, and execution of specific pieces of business logic. Pages generate events (that is, which link or button the customer clicks) that result in the invocation of input processors and Pipelines. These in turn either succeed or generate exceptions, from which the Webflow decides which page to display or which piece of business logic to execute next.

This separation between presentation and business logic is beneficial for a number of reasons, but most importantly, it is helpful from a customization/maintenance standpoint. Different people within your organization may perform different tasks, and may specialize in a particular area. Keeping the user interface separate from the business processes and the Java programming allows your development team to accomplish more in less time, and makes it easier for members of the team to focus on their areas of expertise or interest. For a description of typical roles, see “Development Roles” on page 1-5.

Architecture Categories

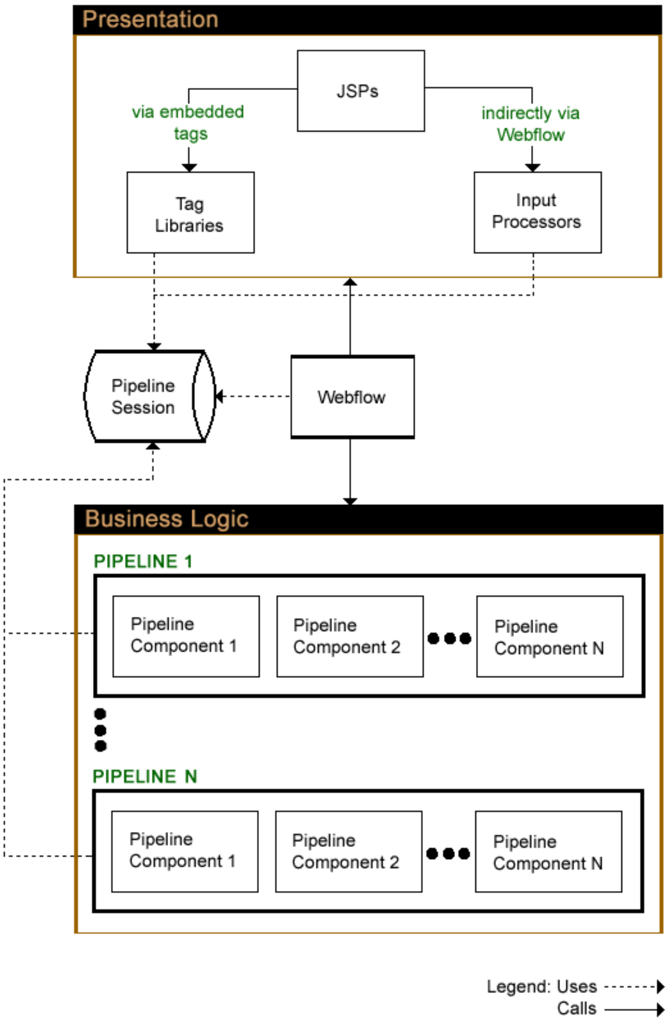
The six technologies previously described can best be understood as belonging to four categories: presentation, business logic, state maintenance, and flow of control.

HTML, JSP tags, and input processors consitute the presentation portion of the system. HTML is the display language understood by most browsers. JSP tags translate information from the Pipeline session to HTML, while input processors translate form data from HTML to the Pipeline session.

The Pipeline components containing pieces of business logic have no knowledge of HTML or any of the other presentation technologies. Instead, the Pipeline session can maintain conversational state in the system. Similarly, the Webflow governs the flow of control.

Figure 1-1 illustrates how the various technology categories interact to preserve the WebLogic Commerce Server design model. Understanding this diagram is essential to understanding how to customize and extend the Webflow and Pipeline mechanisms.

Figure 1-1 Webflow and Pipeline High-level Architecture



As you learn more about the Webflow and Pipeline mechanisms, return to this architecture diagram. Each time you review the diagram, you will have a better understanding of the big picture.

Development Roles

This document is intended for the following audiences:

- The commerce engineer/JSP content developer, who uses JSP templates and tag libraries to implement interactive Web pages to meet business requirements. This user also maintains simple configuration files.
- The business analyst, who defines the company's business protocols (processes and rules) for a business-to-consumer Web site. This user may set pricing policies and discounts, and may plan promotional advertising.
- The site administrator, who uses Commerce and Personalization Server administration screens to configure the site's rules, portals, property sets, user profiles, content delivery, and product catalog.
- The Java/EJB programmer, who creates custom code to insert in the JSP files. This user may also handle complex configuration files and write new Pipeline components or input processors.

Next Steps

The BEA WebLogic Commerce Server product ships with a working e-commerce site that can easily be modified to meet your specific business requirements. Many modifications, such as changes to page layout and presentation, can be completed without any Java coding. It is expected that these changes will be performed by a commerce engineer/JSP content developer or a site administrator, who consults with a business analyst about business strategies. For detailed information about how the Webflow and Pipeline mechanisms work, see Chapter 2, "Customizing Webflow and Pipelines," in this guide. Additionally, Chapter 3, "Using the Webflow and Pipeline Editor," describes how a commerce engineer/JSP developer can use the Webflow and Pipeline Editor Administration Tool to customize and validate the `webflow.properties` and `pipeline.properties` files.

Some of the more complex modifications, such as adding a Pipeline component to use a different credit card authorization system, will require Java coding. It is expected that these changes will be performed by a Java/EJB programmer. For information about how to extend the Webflow and Pipeline, see Chapter 4, “Extending Webflow and Pipelines.”

The WebLogic Commerce Server also contains a few JSP tags specifically designed to work with the Webflow and Pipeline mechanisms. You will use these JSP tags regardless of whether you are customizing or extending the Webflow and Pipeline. For detailed information about how to use the Webflow and Pipeline JSP Tags, see Chapter 5, “Webflow and Pipeline JSP Tags Library Reference.”

CHAPTER

2 Customizing Webflow and Pipelines

The most important benefit of the Webflow and Pipeline mechanisms is that they allow people with different levels of technical skill to customize both the presentation and business logic within an e-commerce site.

Commerce engineers/JSP content developers, site administrators, and business analysts can now divide Web site customization (and subsequent maintenance) activities based on their own expertise, interests, and job responsibilities. While they are working with the Webflow/Pipeline infrastructure, the Java/EJB programmers on the development team can be extending the BEA WebLogic Commerce Server packages to add functionality. Thus, some bottlenecks in the site development and maintenance process are greatly reduced.

If the packages that the BEA WebLogic Commerce Server product provides completely meet your requirements, all you may need to do to have a fully functioning e-business is to customize some aspects of the Webflow and/or Pipeline. This topic describes how to accomplish this.

This topic includes the following sections:

- Using Webflow
  - Customizing Webflow Using the webflow.properties File
  - Using Webflow in Your Web Pages
  - Webflow Search Order
- Using Input Processors with Webflow
  - Syntax of Input Processors in the webflow.properties File
  - Chaining Input Processors

  - Further Customization of Input Processors
- Using Pipelines with Webflow
  - Customizing Pipelines Using the pipeline.properties File
  - Using Pipelines in the Webflow
  - Further Customization of Pipelines

Using Webflow

Since every e-business is different, the BEA WebLogic Commerce Server product utilizes an external properties file to manage the sequence (flow) in which Web pages are displayed. The WebLogic Commerce Server provides a default Webflow properties file to get you up and running quickly, and to provide you with a working example of this concept. You can modify this file to change the order of your pages, without having to edit each page individually.

This section provides information about the default Webflow and instructions for customizing it. This section also describes how to invoke the Webflow mechanism from your Web pages, and explains how missing transitions in the properties file are resolved.

Customizing Webflow Using the webflow.properties File

The Webflow properties file (`webflow.properties`) controls the display of your site's Web pages and initiates execution of the business logic associated with these pages. The Webflow properties file contains one section for each JavaServer Page (JSP) and includes comments for increased readability.

Generically, each transition in the `webflow.properties` file can be written as:

```
<origin>.[<event>][(<eventName>)]=<target>
```

Table 2-1 lists the valid values for each of these elements.

Table 2-1 Valid Values for webflow.properties Elements

Element	Valid Values
`<origin>`	`begin | <page>.<extension> | <inputprocessorName> | <pipelineName>`
`<event>`	`event = link(<linkName>) | button(<buttonName>) | success | exception(<exceptionName>)`
`<target>`	`<page>.<extension> | <inputprocessorName> | <pipelineName>`
`<extension>`	`jsp | html | htm | inputprocessor | pipeline`

Notes: Valid characters for `<page>`, `<inputprocessorName>` and `<pipelineName>` are limited to A-Z, a-z, and an underscore.

Transitions in the `webflow.properties` file should never contain spaces. Including spaces can result in errors that are difficult to locate.

Text within the `webflow.properties` file is case sensitive.

Syntax of the webflow.properties File

Each transition in the `webflow.properties` file is comprised of a name/value pair, separated by an equal sign (=).

The name consists of the current state and a named event, and the value is a result state. In Listing 2-1, the current state is `firstpage.jsp`. The event is a button named Next, and the result state is `nextpage.jsp`.

Listing 2-1 Webflow Properties Example

```
firstpage.jsp.button(next)=nextpage.jsp
```

When a customer clicks the Next button from `firstpage.jsp`, the Webflow will load `nextpage.jsp`.

Note: The only exception to this syntax is the transition in the `webflow.properties` file that defines the initial state for the Webflow, as follows:

```
begin=home.jsp
```

Web pages used as current or result states in the Webflow may be `.htm`, `.html`, or `.jsp` files. In addition to the `button` event shown in the previous example, there is also a `link` event associated with these file types.

About Event Names

Events are given names because it is likely that a page has multiple events of the same type associated with it (that is, there are both previous and next buttons on `firstpage.jsp`, each requiring different result states). Event names are used to differentiate between these events, as shown in Listing 2-2.

Listing 2-2 Event Names Example

```
firstpage.jsp.button(previous)=previouspage.jsp

firstpage.jsp.button(next)=nextpage.jsp
```

Note: Although event names are arbitrarily selected, duplication of names for events of the same type would defeat their purpose and should be avoided. Because duplicate names would produce unpredictable results, the Webflow and Pipeline Editor's validation tool will check for and notify you about any such occurrences. For more information about using the Webflow and Pipeline Editor to validate your `webflow.properties` file, see "How to Validate Your Properties Files" on page 3-58.

Although all the states in the previous examples are JSPs, both current and result states can also be input processors or Pipelines. For more information on input processors and Pipelines, see "Using Input Processors with Webflow" on page 2-10 and "Using Pipelines with Webflow" on page 2-13, respectively.

Using the Wildcard Character

In cases where you want all your Web pages to reach a certain target page, you can substitute the wildcard character (*) for a specific page name in the current state. For example, if you want customers to be able to reach the home page from every page within your Web site, a transition in the `webflow.properties` file would read:

```
*.jsp.link(home)=home.jsp
```

Default Webflow

Listing 2-3 shows the portion of the default Webflow that handles category browsing. It can also be viewed in a simple text editor by opening `WL_COMMERCE_HOME/webflow.properties`, where `WL_COMMERCE_HOME` is the top-level directory where you installed Campaign Manager for WebLogic (which includes WebLogic Commerce Server) or WebLogic Commerce Server.

Listing 2-3 Default Webflow for Category Browsing

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

# Handle category browsing

############################################

# Generic browse link gets browse parameters
*.jsp.link(browse)=BrowseCategory.inputprocessor

# Move intermediate results
BrowseCategory.inputprocessor.success=MoveSiblingResults.inputprocessor

# Get all category detail
MoveSiblingResults.inputprocessor.success=GetBrowseDetails.pipeline

# Display category detail
GetBrowseDetails.pipeline.success=commerce/catalog/browse.jsp

# Handle errors
BrowseCategory.inputprocessor.exception(ProcessingException)=commerce/catalog/
browse.jsp

MoveSiblingResults.inputprocessor.exception(ProcessingException)=commerce/
catalog/browse.jsp

GetBrowseDetails.pipeline.exception(PipelineFatalException)=commerce/catalog/
browse.jsp
```

```
# Define the input processor classes
BrowseCategory.inputprocessor=com.beasys.commerce.ebusiness.catalog.webflow.
GetCategoryIP

MoveSiblingResults.inputprocessor=com.beasys.commerce.ebusiness.catalog.webflow
.MoveAttributeIP
```

Dynamically Modifying Your Site's Webflow

To dynamically modify your site's Webflow, consider the following:

- It is expected that a commerce engineer/JSP content developer (or someone with similar technical knowledge and abilities) will update the `webflow.properties` file.
- Be sure to modify the `webflow.properties` file in your development environment until you achieve the desired outcome. Then move your changes to a production environment.

To modify your site's Webflow, use the Webflow and Pipeline Editor Administration Tool. For more information about this tool, see Chapter 3, "Using the Webflow and Pipeline Editor."

Note: You can also modify the Webflow by editing the `webflow.properties` file directly. However, this is not recommended. Once you edit the `webflow.properties` file by hand, you may not be able to use the Webflow and Pipeline Editor. If you are certain you want to edit the file directly, follow these steps:

1. Start a simple text editor like Notepad.
2. Open the default Webflow properties file, which can be found in `WL_COMMERCE_HOME/webflow.properties`, where `WL_COMMERCE_HOME` is the top-level directory where you installed WebLogic Commerce Server.
3. Modify the file as necessary, using the syntax described in the previous sections.
4. Save the modified file. You do not need to restart the server to view your changes if you have set the `webflow.hotdeploy.enable` property to `true` in the `weblogiccommerce.properties` file.

Using Webflow in Your Web Pages

To utilize the Webflow mechanism, the URLs within your Web pages must include information that corresponds to a transition in the `webflow.properties` file. Specifically, the URL must contain a page name and an event that match a current state in the `webflow.properties` file, as shown in Listing 2-4 and Listing 2-5.

Listing 2-4 URL Within a <FORM> Tag in the Web Page

```
<FORM method=”post”
action=”<%=WebflowJSPHelper.createWebflowURL(pageContext,
“login.jsp”, button(createUser), false)%>”>
```

Listing 2-5 Corresponding Transition in the webflow.properties File

```
login.jsp.button(createUser)=nextpage.jsp
```

These URLs are dynamically generated by a utility class called `WebflowJSPHelper`.

Note: In most cases, a Web page will use the `WebflowJSPHelper` class multiple times. Therefore, it is a good idea to import the class at the beginning of your Web page as shown in the following statement:

```
<%@ page import=”com.beasys.commerce.webflow.WebflowJSPHelper” %>
```

As shown in Listing 2-4, the `WebflowJSPHelper` class has a `createWebflowURL()` method that takes four parameters: `pageContext`, the name of the current JSP with extension (origin), the event type and name, and a URL type. A URL type of `true` causes the returned string to include the origin and event parameters as query parameters. Using the information from these four parameters, the `createWebflowURL()` method returns an absolute URL.

Although the parameters of the `createWebflowURL()`method are always the same, the way you specify these parameters depends on whether you are generating the URL within a `<FORM>` tag or an `<A>` (anchor) tag.

To incorporate a URL using the `<FORM>` tag, use the `action` attribute to construct the URL as shown in Listing 2-4. To incorporate a URL using the `<A>` tag, call the `createWebflowURL()` method as shown in Listing 2-6.

Listing 2-6 Dynamic URL Generation Within an <A> Tag

```
<a href="<%=WebflowJSPHelper.createWebflowURL(pageContext,
"login.jsp", button(createUser), false)%>">
```

In both cases, these statements are translated into `login.jsp.button(createUser)`, which can be found in the `webflow.properties` file to the left of an equal sign, as shown in Listing 2-5. The value to the right of this equal sign initiates the result state, and thus allows the Webflow mechanism to continue.

Webflow Search Order

There may be times when a transition in the Webflow is missing (that is, no result state has been specified). To prevent any problem from being visible to your customer, the Webflow will attempt to resolve missing transitions by searching through several possibilities to locate an alternate flow. These search possiblities are examined by the Webflow mechanism in the following order:

- The Webflow substitutes the wildcard character for the specific page, input processor, or Pipeline.
- If wildcard subsitution fails, the Webflow produces a configuration exception relative to where it encountered the missing transition, and uses this contextual exception as the result state.
- If contextual configuration exceptions do not allow the Webflow to continue, the Webflow combines the wildcard substitution with a generic exception, which it uses as the result state.
- If the previous attempts fail, the Webflow will simply load a configuration error page.

Note: The configuration error page can be specified in the `webflow.properties` file under the property `configurationerrorpage`.

In summary, the search order attempts to prevent a missing transition in the Webflow from interrupting a customer's experience on your Web site. Rather, in the very worst case, the Webflow would load the configuration error page. If for some reason this file was missing, a predefined system error page (`servererror.jsp`), which is also beyond the scope of the Webflow mechanism, would be used instead.

Note: Using the Validate option in the Webflow and Pipeline Editor Administration Tool can help identify potential problems with your Webflow. For more information about the Validate option, see "How to Validate Your Properties Files" on page 3-58.

Search Order Examples

Suppose the Webflow mechanism is attempting to locate the missing transition `login.jsp.link(home)`in the `webflow.properties` file. The following list illustrates the alternate transitions that may be used by the Webflow:

- `*.jsp.link(home)`
- `login.jsp.error(ConfigurationException)`
- `*.jsp.error(ConfigurationException)`
- `configurationerrorpage`

The Webflow search order will also be performed for input processors and Pipelines that are missing in the `webflow.properties` file. The following list illustrates the alternate transitions that may be used by the Webflow for the missing transition `ShoppingCartIP.inputprocessor.success`:

- `*.inputprocessor.success`
- `ShoppingCartIP.inputprocessor.error(ConfigurationException)`
- `*.inputprocessor.error(ConfigurationException)`
- `configurationerrorpage`

Similarly, the following list illustrates the alternate transitions that may be used by the Webflow for the missing transition `ShoppingCartPC.pipeline.success`:

- `*.pipeline.success`
- `ShoppingCartPC.pipeline.error(ConfigurationException)`
- `*.pipeline.error(ConfigurationException)`
- `configurationerrorpage`

Note: For more information about input processors, see “Using Input Processors with Webflow” on page 2-10. For more information about Pipelines, see “Using Pipelines with Webflow” on page 2-13.

Using Input Processors with Webflow

States in the `webflow.properties` file are not restricted to other Web pages. Input processors are predefined classes that provide a way to indirectly carry out more complex tasks using the Webflow mechanism. Input processors reduce the need to incorporate complex Java code into your JSPs, and help maintain the separation between presentation and business logic.

The role of input processors is to read data from the `HTTPServletRequest` and use it to create or update Java objects in a Pipeline session. In addition to working with this data, some input processors may also validate information supplied by the customer.

Note: It is not required that you use input processors in your customized Webflow. If you do not wish to use input processors, simply do not specify any input processors in the `webflow.properties` file.

This section provides information about invoking input processors from theWebflow and about chaining input processors. This section also points you to additional information about extending or developing your own input processors.

Syntax of Input Processors in the webflow.properties File

Input processors extend the syntax used for JSPs in the `webflow.properties` file. For example, if you want to verify that the customer filled in the required form fields for their address before sending the customer to the next page, you could use the `ValidateAddress` input processor as shown in Listing 2-7.

Listing 2-7 Input Processor for Address Validation

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

# ValidateAddress input processor

#################################

# Invoke the input processor
addaddress.jsp.button(continue)=ValidateAddressIP.inputprocessor

# Specify the fully qualified class name for the input processor
ValidateAddressIP.inputprocessor=com.beasys.commerce.ebusiness.
customer.webflow.ValidateAddressIP

# Specify the result state for successful execution
ValidateAddressIP.inputprocessor.success=selectaddress.jsp

# Specify the result state for unsuccessful execution
ValidateAddressIP.inputprocessor.exception(ProcessingException)=
addaddress.jsp
```

In the first transition, a customer who clicks the Continue button causes the flow to be turned over to the input processor called `ValidateAddressIP`. The second line defines the full class name of the `ValidateAddressIP` input processor, which will validate the form fields. The transitions that follow make use of the event types defined for input processors: `success` and `exception`. If the validation is successful, the result state indicated by the success event is to load the `selectaddress.jsp` file. If the validation is not successful, the `ValidateAddressIP` input processor directs the customer back to `addaddress.jsp` to make corrections.

Notes: For the complete list of event types and more information on the syntax of input processors in the Webflow, see Table 2-1.

If execution of an input processor is not successful, you may specify different result states identified by more than one exception event.

Chaining Input Processors

In addition to using input processors between JSPs and Pipelines, you can also use more than one input processor, or chain input processors. In a chaining arrangement, the result state of one successfully executed input processor will be another input processor, as shown in Listing 2-8.

Listing 2-8 Example of Input Processor Chaining

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

# Example of input processor chaining

#####################################

# Invoke the first input processor
webpage.jsp.link(continue)=firstInputProcessor

# Specify the fully qualified path name for the first input
# processor
firstInputProcessor.inputprocessor=com.beasys.commerce.webflow.
firstInputProcessorIP

# Invoke the second input processor if the execution of the first
# input processor succeeds
firstInputProcessor.inputprocessor.success=secondInputProcessor

# Specify the fully qualified class name for the second input
# processor
secondInputProcessor.inputprocessor=com.beasys.commerce.webflow.
secondInputProcessorIP

# Specify the result state for successful execution of the second
# input processor
secondInputProcessor.inputprocessor.success=nextwebpage.jsp

# Specify the result state for unsuccessful execution of the first
# input processor
firstInputProcessor.inputprocessor.exception(ProcessingException)
=errorpage.jsp

# Specify the result state for successful execution of the second
# input processor
secondInputProcessor.inputprocessor.exception
(ProcessingException)=anothererrorpage.jsp
```

Further Customization of Input Processors

If you would like to customize your site even further, you might choose to create and implement your own input processors or define your own exceptions for use with input processors. However, there are some important rules you need to follow to accomplish these tasks. For more information, see Chapter 4, “Extending Webflow and Pipelines.”

Note: Only Java/EJB programmers (or someone with similar technical knowledge and abilities) should attempt to customize input processors.

Using Pipelines with Webflow

Your site would not be considered an e-business if you simply displayed pages and performed some additional tasks with input processors. A customer’s entire experience also relies upon the execution of back-end business processes that are related to where the customer is on your site and what the customer is trying to accomplish.

A Pipeline is an advanced mechanism invoked by the Webflow that initiates execution of specific tasks related to your business process. For example, if a customer attempts to move to another page on your site but you want to save the customer’s identification information to a database first, you could use a Pipeline.

All Pipelines are collections of individual Pipeline components, which can be implemented as Java objects or stateless session EJBs. Pipeline components are the parts of a Pipeline that actually perform the tasks associated with the underlying business logic. When these tasks are complex, Pipeline components may also make calls to external services (other business objects). As in the case of input processors, the BEA Weblogic Commerce Server product provides predefined Pipeline components that you can use, or you can customize your site further by creating your own.

To successfully carry out business tasks, each Pipeline component must read attributes from a Pipeline session and if necessary, write modified versions of these attributes back to the Pipeline session. By default, attributes in the Pipeline session are available for the life of the HTTP session.

The WebLogic Commerce Server provides a default Pipeline properties file to get you up and running quickly, and to provide you with a working example of this concept. You can modify this file to change the business logic associated with your Web pages, without having to edit each page individually.

Note: It is not required that you use Pipelines to execute business logic in your customized Webflow. If you do not wish to use Pipelines, simply do not specify any Pipelines in the `webflow.properties` file. However, eliminating Pipelines and Pipeline components results in a less scalable, 2-tier architecture instead of the 3-tier architecture provided by the Webflow/Pipeline infrastructure.

This section provides information about the default Pipeline and instructions for customizing it, and describes how to invoke Pipelines from the Webflow.

Customizing Pipelines Using the pipeline.properties File

Much like the `webflow.properties` file specifies the flow of Web pages presented to a customer, the Pipeline properties file (`pipeline.properties`) specifies the flow of business logic as the customer moves through each page of the site. This properties file contains one section for each JavaServer Page (JSP) and includes comments for increased readability.

Generically, Pipeline definitions can be written as:

```
<pipelineName>.componentList
<pipelineName>.isTransactional=<true|false>
```

where `componentList` is a comma-separated list of Pipeline components to be executed in sequence.

Once all Pipeline definitions are complete, you must specify definitions for each Pipeline component in the Pipeline. Each Pipeline component definition consists of three properties: `className`, `jndiName`, and `isEJBSessionBean`.

Table 2-2 describes each of the Pipeline component properties in detail.

Table 2-2 Pipeline Component Properties

Property	Description	Value
`className`	Name of the class that implements the Pipeline component, required if `isEJBSessionBean=false`	A fully qualified Java class name
`isEJBSessionBean`	Specifies whether or not the Pipeline component is a session bean, and always requires a value	`true|false`
`jndiName`	JNDI name of the session bean that implements the Pipeline component, required only if `isEJBSessionBean=true`	A string

Notes: Lines in the `pipeline.properties` file should never contain spaces. Including spaces can result in errors that are difficult to locate.

Text within the `pipeline.properties` file is case sensitive.

Syntax of the pipeline.properties File

The top portion of the `pipeline.properties` file should contain only Pipeline definitions. Pipeline definitions include:

- A Pipeline name.
- A list of its associated Pipeline components in order of execution.
- A value for the `isTransactional` Pipeline property, indicating whether or not all the Pipeline components in the Pipeline will participate in a transaction.

Listing 2-9 is a Pipeline definition that might be used in the `pipline.properties` file.

Listing 2-9 Pipeline Definition Example

```
orderPipeline=CalculateTaxPC,CalculateDiscountPC,TotalCartCostPC
orderPipeline.isTransactional=true
```

In this example, the Pipeline called `orderPipeline` consists of three Pipeline components (`CalculateTaxPC, CalculateDiscountPC, TotalCartCostPC`). The `orderPipeline` is also transactional.

Listing 2-10 shows the corresponding Pipeline component definitions that might be used in the `pipeline.properties` file.

Listing 2-10 Pipeline Component Definition Example

```
CalculateTaxPC.classname=com.beasys.commerce.ebusiness.order.
pipeline.CalculateTaxPC
CalculateTaxPC.isEJBSessionBean=false
CalculateTaxPC.jndiName=

CalculateDiscountPC.classname=com.beasys.commerce.ebusiness.order
.pipeline.CalculateDiscountPC
CalculateDiscountPC.isEJBSessionBean=false
CalculateDiscountPC.jndiName=

TotalCartCostPC.classname=com.beasys.commerce.ebusiness.order
pipeline.TotalCartCostPC
TotalCartCostPC.isEJBSessionBean=true
TotalCartCostPC.jndiName=com.beasys.commerce.ebusiness.order.
pipeline.TotalCartCostPC
```

Default Pipeline

Listing 2-11 shows portions of the default Pipeline property file that handle obtaining product categories (with Pipeline components implemented as Java objects) and moving an item to a shopping cart (with a Pipeline component implemented as an EJB session bean). These can also be viewed in a simple text editor by opening `WL_COMMERCE_HOME/pipeline.properties`, where `WL_COMMERCE_HOME` is the top-level directory where you installed WebLogic Commerce Server.

Listing 2-11 Default Pipelines for Product Categories and Shopping Cart

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

# Java class Pipeline for obtaining product categories

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

```
# GetTopCategories Pipeline definition
GetTopCategories.componentList=GetCategoryPC,GetSubcategoriesPC
GetTopCategories.isTransactional=false

# GetCategoryPC Pipeline component definition
GetCategoryPC.classname=com.beasys.commerce.ebusiness.catalog.
pipeline.GetCategoryPC
GetCategoryPC.jndiName=
GetCategoryPC.isEJBSessionBean=false

# GetSubcategoriesPC Pipeline component definition
GetSubcategoriesPC.classname=com.beasys.commerce.ebusiness.
catalog.pipeline.GetSubcategoriesPC
GetSubcategoriesPC.jndiName=
GetSubcategoriesPC.isEJBSessionBean=false

#############################################################

# EJB session bean Pipeline for moving items to shopping cart

#############################################################

# MoveProductItemToShoppingCart Pipeline definition
MoveProductItemToShoppingCart.componentList=
MoveProductItemToShoppingCartPC
MoveProductItemToShoppingCart.isTransactional=true

# MoveProductItemToShoppingCartPC Pipeline component definition
MoveProductItemToShoppingCartPC.classname=com.beasys.commerce.
ebusiness.shoppingcart.pipeline.MoveProductItemToShoppingCartPC
MoveProductItemToShoppingCartPC.jndiName=com.beasys.commerce.
ebusiness.shoppingcart.pipeline.MoveProductItemToShoppingCartPC
MoveProductItemToShoppingCartPC.isEJBSessionBean=true
```

Dynamically Modifying Your Site's Pipelines

To dynamically modify your site's Pipelines, consider the following:

- It is expected that a business analyst will work with the commerce engineer/JSP content developer (or someone with similar technical knowledge and abilities) to update the `pipeline.properties` file.
- Be sure to modify the `pipeline.properties` file in your development environment until you achieve the desired outcome. Then move your changes to a production environment.

To modify your site's Pipelines, use the Webflow and Pipeline Editor Administration Tool. For more information about this tool, see Chapter 3, "Using the Webflow and Pipeline Editor."

Note: You can also modify Pipelines by editing the `pipeline.properties` file directly. However, this is not recommended. Once you edit the `pipeline.properties` file by hand, you may not be able to use the Webflow and Pipeline Editor. If you are certain you want to edit the file directly, follow these steps:

1. Start a simple text editor like Notepad.
2. Open the default Pipeline properties file, which can be found in `WL_COMMERCE_HOME/pipeline.properties`, where `WL_COMMERCE_HOME` is the top-level directory where you installed WebLogic Commerce Server.
3. Modify the file as necessary, using the syntax described in the previous sections.
4. Save the modified file. You do not need to restart the server to view your changes if you have set the `pipeline.hotdeploy.enable` property to `true` in the `weblogiccommerce.properties` file.

Eliminating Pipeline Components

The `pipeline.properties` file provides an easy way to eliminate Pipeline components without the need for advanced programming skills. For example, you might want to eliminate a Pipeline component that performs your tax calculations in the `CommitOrder` Pipeline. The definition in the default `pipeline.properties` file for the `CommitOrder` Pipeline is shown in Listing 2-12.

Listing 2-12 Default CommitOrder Pipeline

```
# CommitOrder
CommitOrder.componentList=CommitOrderPC, AuthorizePaymentPC,
TaxCalculateAndCommitLineLevelPC
```

To eliminate a Pipeline component using a text editor like Notepad, follow these steps:

Notes: Hand-editing the `pipeline.properties` file is not recommended. Instead, you should utilize the Webflow and Pipeline Editor tool, described in Chapter 3, "Using the Webflow and Pipeline Editor." These instructions are simply intended as an example for understanding how the content of the Pipeline configuration file is structured.

These instructions assume that the WebLogic Commerce Server software is installed and has been started.

You do not need to restart the server to view your changes if you have set the `pipeline.hotdeploy.enable` property to `true` in the `weblogiccommerce.properties` file.

1. Open the `pipeline.properties` file and remove each reference to the Pipeline component you want to eliminate. For example, if tax calculations are not required, remove all tax calculation Pipeline components from the Pipeline definition, as shown in Listing 2-13. Be sure to save your changes.

Listing 2-13 CommitOrder Pipeline Without Tax Calculation Component

```
# CommitOrder
CommitOrder.componentList=CommitOrderPC, AuthorizePaymentPC
```

2. If necessary, edit the related JSPs to eliminate places in the user interface where the information is gathered. Be sure to save your changes.

Note: There is no JSP that collects tax information. In the `CommitOrder` Pipeline example, you would not need to make changes to the user interface.

3. If in step 2 you removed an entire JSP, you will also need to modify the `webflow.properties` file and change any reference(s) to bypass it. Again, this should be done using the Webflow and Pipeline Editor tool, but can technically still be customized by hand.

Reordering Pipeline Components

The `pipeline.properties` file also provides an easy way for you to modify the sequence of Pipeline components, without the need for advanced programming skills.

To reorder a Pipeline component using a text editor like Notepad, all you need to do is open the `pipeline.properties` file and change the order that the Pipeline components are listed in the Pipeline definition. Be sure to save your changes.

Notes: Hand-editing the `pipeline.properties` file is not recommended. Instead, you should utilize the Webflow and Pipeline Editor tool, described in Chapter 3, “Using the Webflow and Pipeline Editor.” These instructions are simply intended as an example for understanding how the content of the Pipeline configuration file is structured.

Using the same `CommitOrder` Pipeline example, say you wanted to authorize the payment after calculating the tax instead of before it (as in the default `OrderCommit` Pipeline). Listing 2-14 shows how to do this.

Listing 2-14 CommitOrder Pipeline with Tax Component Reordered

```
# CommitOrder
CommitOrder.componentList=CommitOrderPC,
TaxCalculateAndCommitLineLevelPC, AuthorizePaymentPC
```

Note: You do not need to restart the server to view your changes if you have set the `pipeline.hotdeploy.enable` property to `true` in the `weblogiccommerce.properties` file.

Using Pipelines in the Webflow

Pipelines are used in the `webflow.properties` file to initiate execution of the business logic required for a particular page. Each Pipeline must first be invoked by the Webflow, and then followed by a success and exception path. For example, if a customer were to submit their order for processing, the `orderPipeline` might be represented in the `webflow.properties` file as shown in Listing 2-15.

Listing 2-15 Using a Pipeline in the Webflow

```
shoppingcart.jsp.button(submit)=orderPipeline.pipeline
orderPipeline.pipeline.success=commitorder.jsp
orderPipeline.pipeline.exception(PipelineFatalException)=shoppingcart.jsp
```

The first transition indicates that when a customer clicks on the Submit button, the Webflow will turn control over to the Pipeline called `orderPipeline`. If the Pipeline executes successfully (that is, if each component in the Pipeline executes without error), the second transition sends the customer to a page that allows the customer to commit the order. If the Pipeline does not execute successfully, the third transition specifies the exception and directs the customer back to the shopping cart page.

Further Customization of Pipelines

If you would like to customize your site even further, you might choose to create and implement your own Pipelines or define your own exceptions for use with Pipelines. However, there are some important rules you need to follow to accomplish these tasks. For a more information, see Chapter 4, “Extending Webflow and Pipelines.”

Note: Only Java/EJB programmers (or someone with similar technical knowledge and abilities) should attempt to customize Pipelines.

CHAPTER

3 Using the Webflow and Pipeline Editor

The Webflow and Pipeline Editor is a JSP-based administration tool specifically designed to help you modify and validate the `webflow.properties` and `pipeline.properties` configuration files. This topic describes how to access the Webflow and Pipeline Editor, and provides instructions for its use.

This topic includes the following sections:

- About the Webflow and Pipeline Editor
- Starting the WebLogic Commerce Server Administration Tools
- Global Origins
- Page Origins
- Inputprocessor Origins
- Pipeline Origins
- How to Validate Your Properties Files

About the Webflow and Pipeline Editor

The following list contains useful information for site administrators who are about to use the Webflow and Pipeline Editor:

- The Webflow and Pipeline features are available if you have downloaded or purchased the full Campaign Manager for WebLogic licence, or the WebLogic Commerce Server license. (The Campaign Manager for WebLogic license includes the campaign service plus the WebLogic Commerce Server features and WebLogic Personalization Server features.) The Webflow and Pipeline features are not available if you are using the license for WebLogic Personalization Server only.
- The Webflow and Pipeline Editor is designed to help you modify and validate either the `webflow.properties` or `pipeline.properties` files. It does not support the editing of arbitrary files or other WebLogic Commerce Server properties files (such as `weblogiccommerce.properties`).
- The Webflow and Pipeline properties files are located in the root directory of the server where WebLogic Commerce Server is installed. For example, if `WL_COMMERCE_HOME` is the directory where you installed the WebLogic Commerce Server, this is where you will find the `webflow.properties` and `pipeline.properties` files.
- The Webflow and Pipeline Editor should not be used when properties files are also modified by hand (that is, using a text editor such as Notepad). The preferred editing method for these files is through the Webflow and Pipeline Editor. If you create invalid entries outside of the Webflow and Pipeline Editor, the behavior of the editor may be unpredictable. The Webflow and Pipeline Editor will only work with a valid Webflow or Pipeline properties file.
- The Webflow and Pipeline Editor does not support multiple users, and will not detect this behavior. Therefore, your organization must ensure that only one site administrator is using the Webflow and Pipeline Editor at any given time. Failure to do so can result in loss of changes or invalid Webflows.
- The Webflow and Pipeline Editor does not support role-based security. Any user with access to other Administration Tools can access the Webflow and Pipeline Editor tool and vice versa.
- If you are using a cluster without a shared file system, the Webflow and Pipeline Editor will only update the property files associated with the server you are currently logged into. In such cases, BEA recommends editing the Webflow on a staging server and hand-copying the files to other servers as necessary.
- By default, changes made to the Webflow and Pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on Webflow's hot-deploy feature. This can be done in

the `weblogiccommerce.properties` file with the `webflow.hotdeploy.enable` and `pipeline.hotdeploy.enable` properties.

Starting the WebLogic Commerce Server Administration Tools

Before you can use the Webflow and Pipeline Editor, you need to start the server and load the WebLogic Commerce Server Administration Tools page in your Web browser.

To start the server on a Windows system, you can either:

- Run `StartCommerce.bat` from the command line in the `WL_COMMERCE_HOME` directory, where `WL_COMMERCE_HOME` is the directory where you installed the WebLogic Commerce Server.
- From the Start menu, select Programs → BEA WebLogic E-Business Platform → BEA WebLogic Commerce Server 3.5 → Start BEA WebLogic Commerce Server.

To start the server on a UNIX system, run `StartCommerce.sh` from the command line in the `WL_COMMERCE_HOME` directory, where `WL_COMMERCE_HOME` is the directory where you installed the WebLogic Commerce Server.

The Administration Tools page (shown in Figure 3-1) is an entry page into all of the available WebLogic Commerce Server Administration Tools, including the Webflow and Pipeline Editor. To load this page, use one of the following methods:

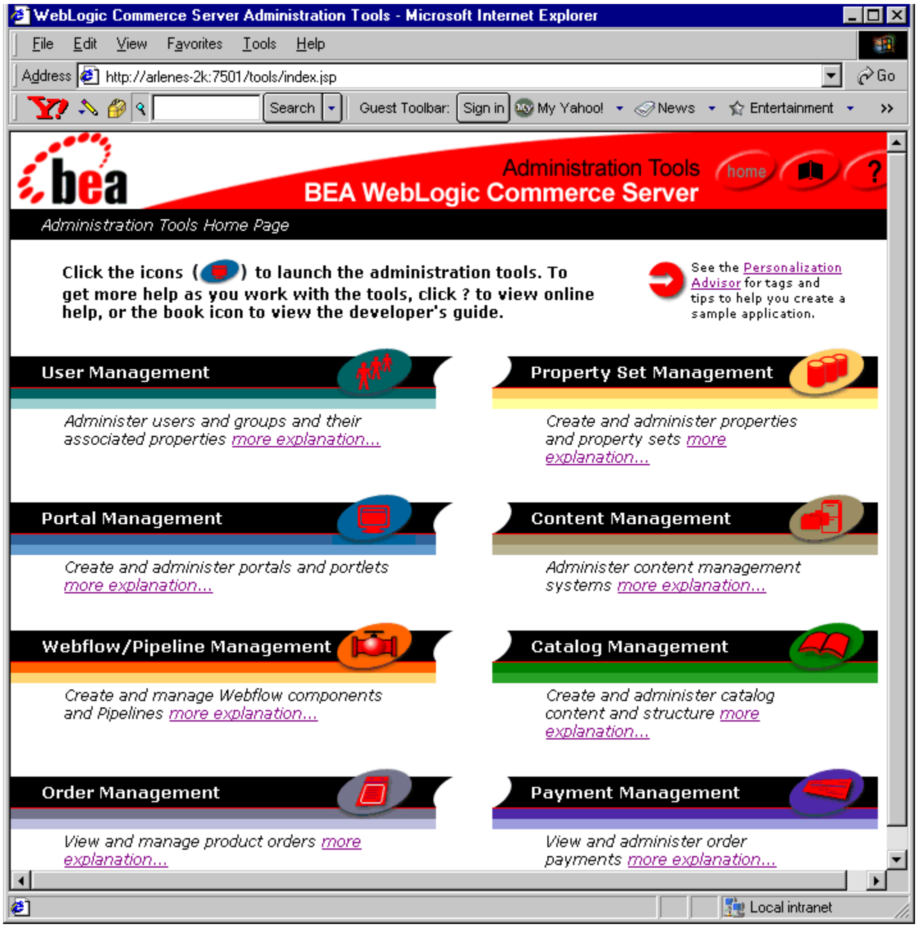
- Specify the URL for the page (http://localhost:7501/tools) in your Web browser.

Note: If you need to perform an administrative task on another node in the cluster, specify the machine such as http://elvis:7501/tools.

- From the Start menu on a Windows system, select Programs → BEA WebLogic E-Business Platform → BEA WebLogic Commerce Server 3.5 → Administration Tools.

You will be asked to supply the username and password for the Administration Tools home page, which will provide you with access to all the administration tools, including the Webflow and Pipeline Editor. The default `username` is `administrator`, and the default `password` is `password`.

Figure 3-1 WebLogic Commerce Server Administration Tools


WebLogic Commerce Server Administration Tools - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://arlenes-2k:7501/tools/index.jsp
Go
Search
Guest Toolbar: Sign in
My Yahoo!
News
Entertainment
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Administration Tools Home Page
Local intranet

Click the icons ( ) to launch the administration tools. To get more help as you work with the tools, click ? to view online help, or the book icon to view the developer's guide.

See the Personalization Advisor for tags and tips to help you create a sample application.

User Management
Administer users and groups and their associated properties more explanation...

Property Set Management
Create and administer properties and property sets more explanation...

Portal Management
Create and administer portals and portlets more explanation...

Content Management
Administer content management systems more explanation...

Webflow/Pipeline Management
Create and manage Webflow components and Pipelines more explanation...

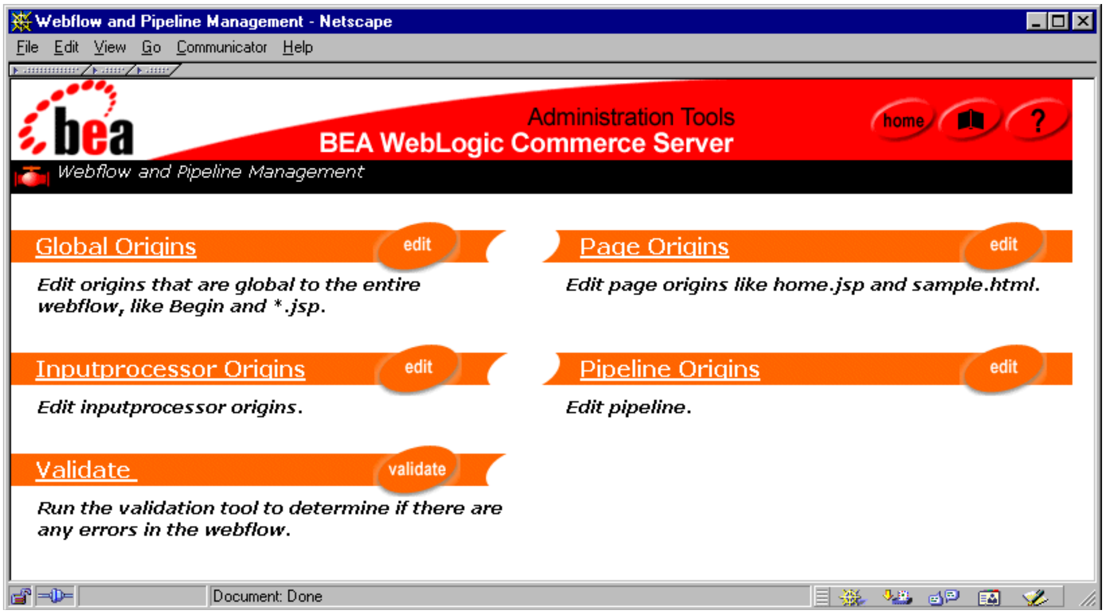
Catalog Management
Create and administer catalog content and structure more explanation...

Order Management
View and manage product orders more explanation...

Payment Management
View and administer order payments more explanation...

To load the Editor, click the pipe icon shown on the Webflow/Pipeline Management title bar. The Webflow and Pipeline Management page appears, as shown in Figure 3-2.

Figure 3-2 Webflow and Pipeline Management


Webflow and Pipeline Management - Netscape

File Edit View Go Communicator Help

Administration Tools

BEA WebLogic Commerce Server

Webflow and Pipeline Management

Global Origins — edit

*Edit origins that are global to the entire webflow, like Begin and *.jsp.*

Page Origins — edit

Edit page origins like home.jsp and sample.html.

Inputprocessor Origins — edit

Edit inputprocessor origins.

Pipeline Origins — edit

Edit pipeline.

Validate — validate

Run the validation tool to determine if there are any errors in the webflow.

Document: Done

From the Webflow and Pipeline Management page, you can create, modify, and remove global origins, page origins, inputprocessor origins, and Pipeline origins. Or, you can choose to validate your Webflow.

Subsequent sections of this document provide instructions about how to perform these tasks using the Webflow and Pipeline Editor.

Global Origins

Recall from Chapter 2, “Customizing Webflow and Pipelines,” that each transition in the `webflow.properties` file contains a current state and a result state separated by an equal sign. The current state is considered an origin, because it represents the page, input processor, or Pipeline where the transition originated. A global origin is an origin that can be used throughout the entire Webflow.

There are seven global origins that may be present in a Webflow. They are:

- *.htm
- *.html
- *.inputprocessor
- *.jsp
- *.pipeline
- begin
- configurationerrorpage

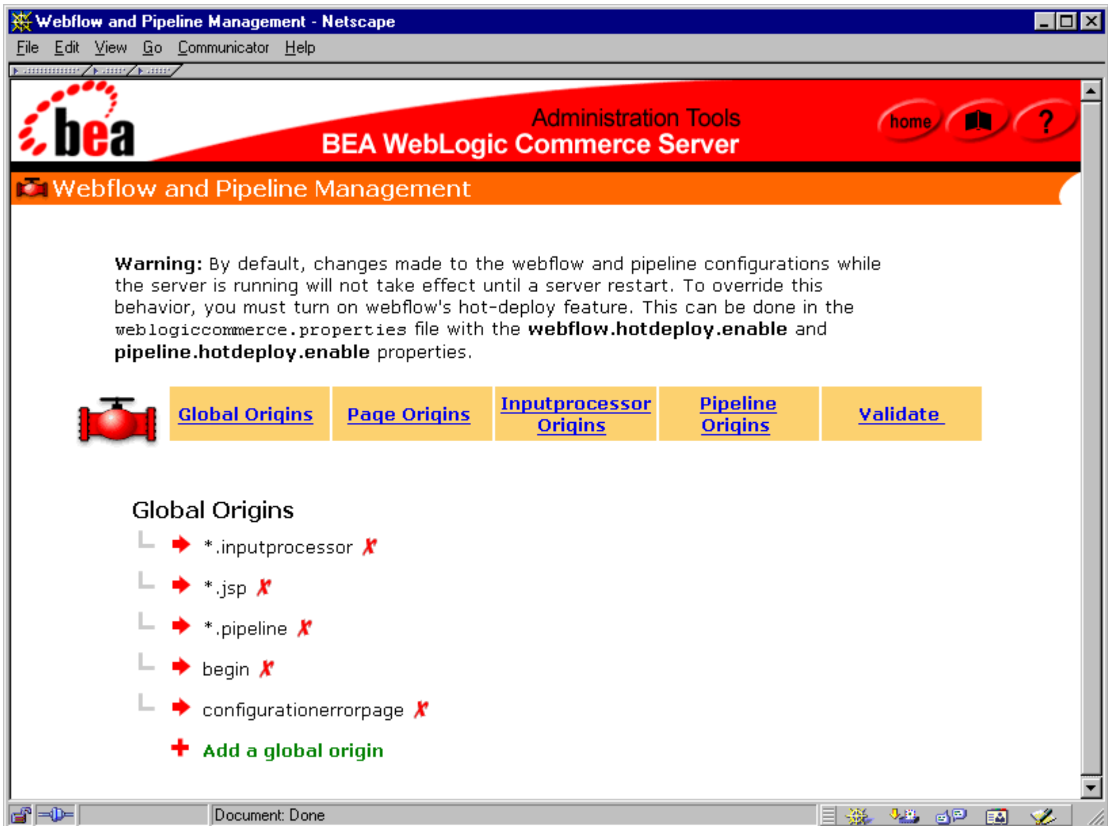
Five of the seven global origins make use of the wildcard character, which indicates that its associated transition is to be performed regardless of the current page, input processor, or Pipeline. For more information about using the wildcard character in the Webflow, see "Using the Wildcard Character" on page 2-5.

The begin origin, which specifies the initial state for the Webflow, is also considered a global origin. For more information about the begin origin, see "Syntax of the webflow.properties File" on page 2-3.

The configuration error page (`configurationerrorpage.jsp`), which is displayed if a transition is missing and the Webflow search order cannot rectify it, is the last global origin. For more information about the configuration error page, see "Webflow Search Order" on page 2-8.

To work with global origins, click the Global Origins link (or its corresponding Edit button). The Global Origins page appears, as shown in Figure 3-3.

Figure 3-3 Global Origins


Webflow and Pipeline Management - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow and Pipeline Management
Global Origins
Page Origins
Inputprocessor Origins
Pipeline Origins
Validate
Document: Done

Warning: By default, changes made to the webflow and pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on webflow's hot-deploy feature. This can be done in the `weblogiccommerce.properties` file with the **webflow.hotdeploy.enable** and **pipeline.hotdeploy.enable** properties.

Global Origins

- *.inputprocessor
- *.jsp
- *.pipeline
- begin
- configurationerrorpage

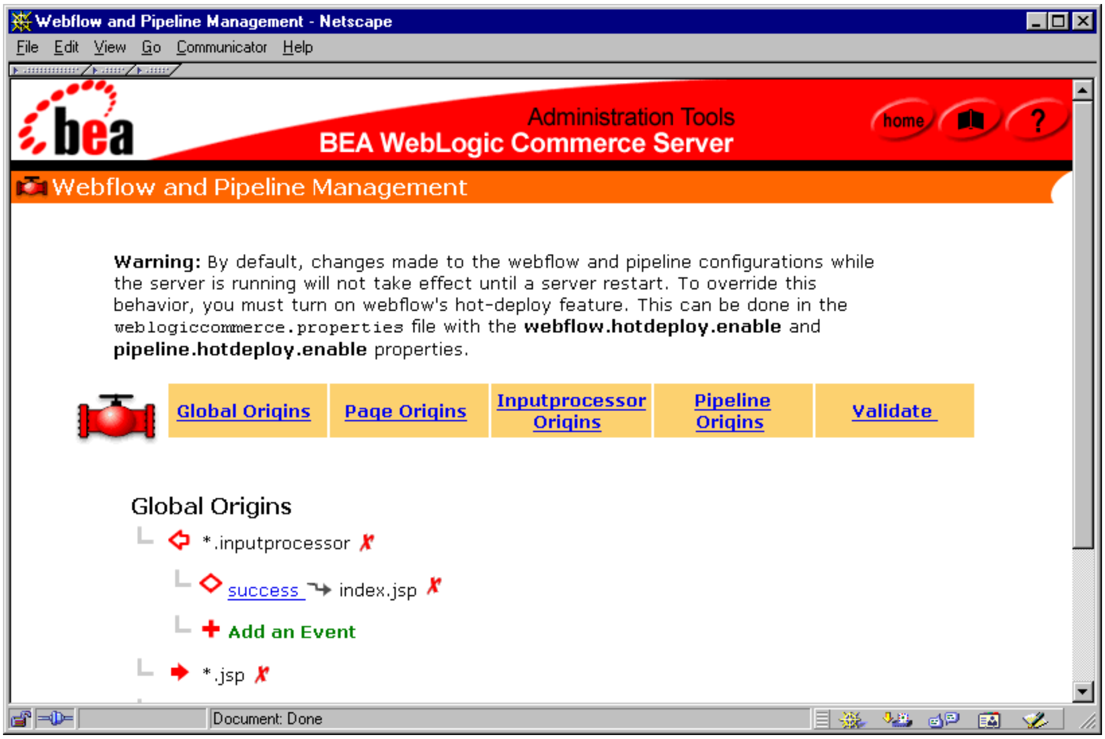
Add a global origin

From this page, you can view, add, and delete global origins. You can also edit global origins by modifying their associated events or by associating new events.

Viewing Global Origins

Before working with global origins, you may first want to see what global origins are currently in the Webflow. When the Global Origins page is first loaded into your browser, you will see a list of the current global origins, but no details about them. To view a global origin's associated events, click the solid arrow located to the left of the global origin. The arrow changes to an open arrow, and the events currently associated with the global origin appear with an open diamond, as shown in Figure 3-4.

Figure 3-4 Viewing Events for the Inputprocessor Global Origin


Webflow and Pipeline Management - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow and Pipeline Management
Global Origins
Page Origins
Inputprocessor Origins
Pipeline Origins
Validate
Add an Event
Document: Done

Warning: By default, changes made to the webflow and pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on webflow's hot-deploy feature. This can be done in the `weblogiccommerce.properties` file with the **webflow.hotdeploy.enable** and **pipeline.hotdeploy.enable** properties.

Global Origins

*.inputprocessor

success → index.jsp

*.jsp

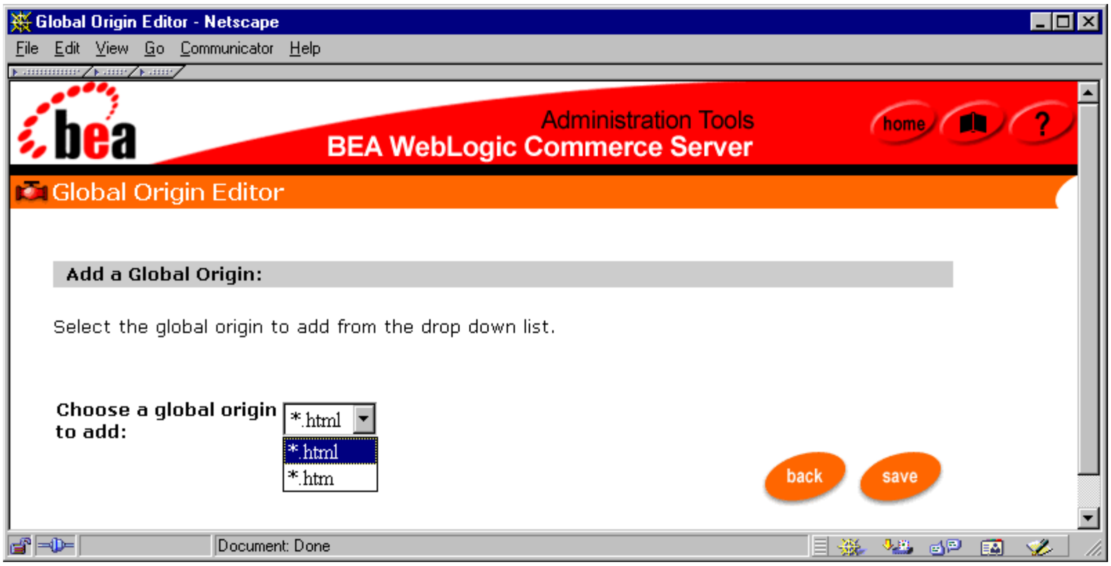
Note: The arrows to the left of each global origin are solid and point toward the origin if it can be expanded to show associated events. If the event information associated with a global origin can be collapsed, the arrow is open and points away from the origin. The `*.jsp` global origin in Figure 3-4 is an example of an origin that can be expanded; the `*.inputprocessor` global origin is an example of an origin that can be collapsed. Events are always represented by open diamonds.

Adding a Global Origin

To add a global origin, follow these steps:

1. Click the plus (+) symbol located to the left of Add a Global Origin. (If the list of global origins is long, you may need to use the scrollbar.) The Add Global Origin page appears, as shown in Figure 3-5.

Figure 3-5 Add Global Origin



Global Origin Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Global Origin Editor
back
save
Document: Done

Add a Global Origin:

Select the global origin to add from the drop down list.

Choose a global origin to add: *.html

*.html
*.htm

2. Select the type of origin (`*.jsp, *.htm, *.html, begin, configurationerrorpage`) you want to add from the Global Origin pull-down menu.

Note: You can only have one global origin per type (that is, one `*.htm`, `*.html`, `*.inputprocessor`, `*.jsp`, `begin`, and `configurationerrorpage`) in a `webflow.properties` file. If there are already some global origins present in the Webflow, only the remaining options will appear in the Global Origin pull-down menu.

3. Click the Save button to save the new global origin. You are returned to the Global Origins page, and your new origin appears alphabetically in the list.
4. Click the solid arrow located to the right of the new global origin to view its associated events. The arrow changes to an open arrow pointing in the opposite direction. A default event (marked by an open diamond) and an option to Add an Event are shown.

Note: For Web pages (`*.htm`, `*.html`, and `*.jsp`), the default event of `link(home)`→ `index.jsp` is shown; For input processors and Pipelines, the default event is `success()`→ `index.jsp`.

5. If you want to add a different event to the global origin, modify the default event, or delete the default event, proceed to the appropriate section in "Editing a Global Origin" below.

Editing a Global Origin

Editing a global origin is synonymous with modifying its associated events. You can add, edit, or delete the events associated with a global origin as described in the following sections.

Adding an Event

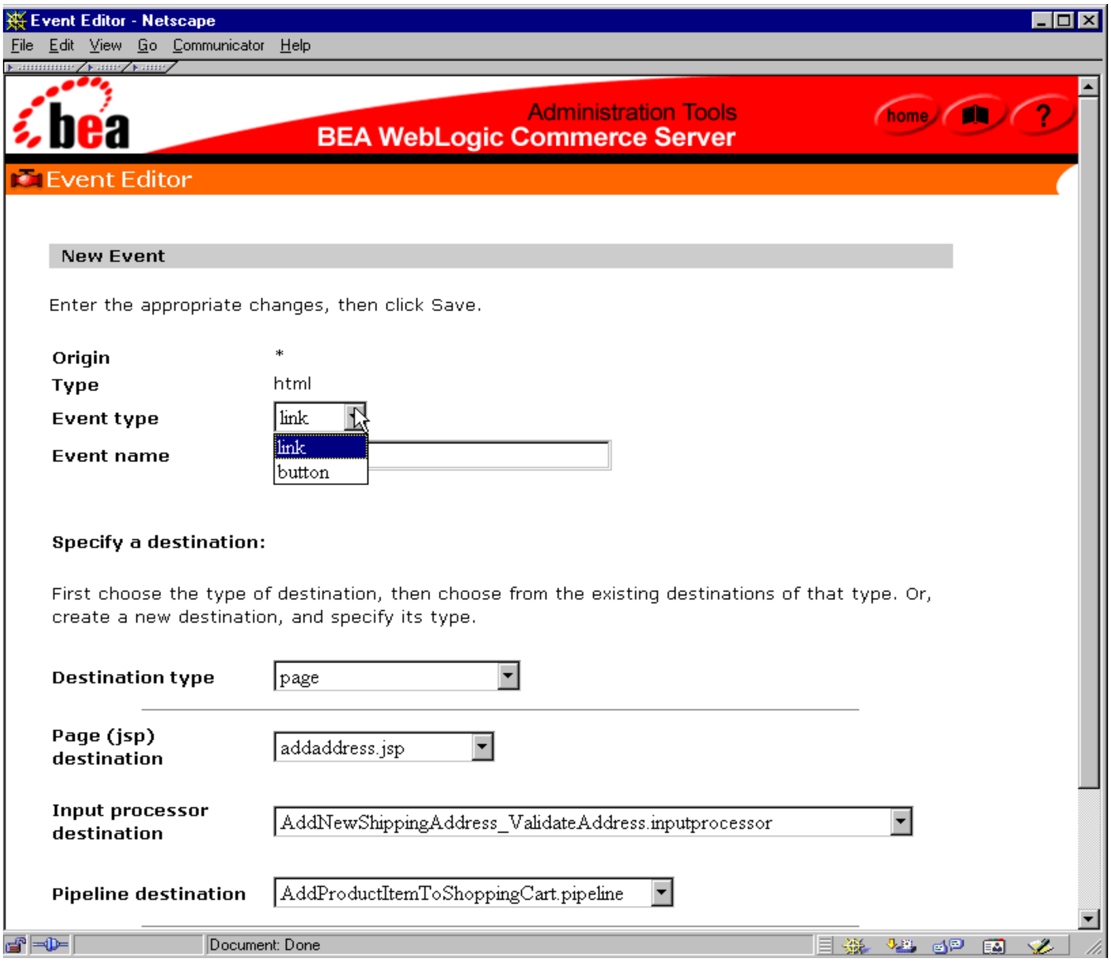
Note: For the `begin` and `configurationerrorpage` global origins, you can only have one event.

To add a new event to a global origin, follow these steps:

1. Verify that the global origin to which you want to add the event is expanded (that is, an open arrow is shown to the left of the global origin, and the Add an Event option is shown). If the global origin is not expanded, click the solid arrow located to the left of the global origin to expand it.

2. Click the plus (+) sign located next to Add an Event. (If the list of events is long, you may need to use the scrollbar.) The New Event page appears, as shown in Figure 3-6.

Figure 3-6 Global Origin: New Event



New Event

Enter the appropriate changes, then click Save.

Origin	*
Type	html
Event type	link (link, button)
Event name	

Specify a destination:

First choose the type of destination, then choose from the existing destinations of that type. Or, create a new destination, and specify its type.

Destination type	page
Page (jsp) destination	addaddress.jsp
Input processor destination	AddNewShippingAddress_ValidateAddress.inputprocessor
Pipeline destination	AddProductItemToShoppingCart.pipeline

3. Select the appropriate Event Type from the pull-down menu, and enter a name for the event in the Event Name text field.

Note: For more information about event types and event names, see “Syntax of the webflow.properties File” on page 2-3 and “About Event Names” on page 2-4, respectively.

4. Next, you need to specify a destination for the event. The destination could be an existing page (`.htm`, `.html`, or `.jsp`), input processor, or Pipeline, or the destination could be an entirely new page, input processor, or Pipeline. To specify the destination, follow these steps:

   a. Select the destination type from the Destination Type pull-down menu (one of Page, Input Processor, Pipeline, or Specify New Destination).

Note: Selecting a destination type in this menu disables form fields that are not required for the chosen destination type. To enable unavailable form fields, choose a different destination type.

   b. If you selected Page from the Destination Type pull-down menu, select an existing page (`.htm`, `.html`, or `.jsp`) from the Page (JSP) Destination pull-down menu. If you selected Input Processor, select an existing input processor from the Input Processor Destination pull-down menu. If you selected Pipeline, select an existing Pipeline from the Pipeline Destination pull-down menu. If you selected Specify New Destination from the Destination Type pull-down menu, enter the name of the destination in the Specify New Destination text field, and select the destination type from the Of Type pull-down menu (jsp, htm, html, inputprocessor, or pipeline).

   c. Click the Save button to save the new event. You are returned to the Global Origins page, and the new event (marked by an open diamond) appears beneath the appropriate global origin in an alphabetic list.

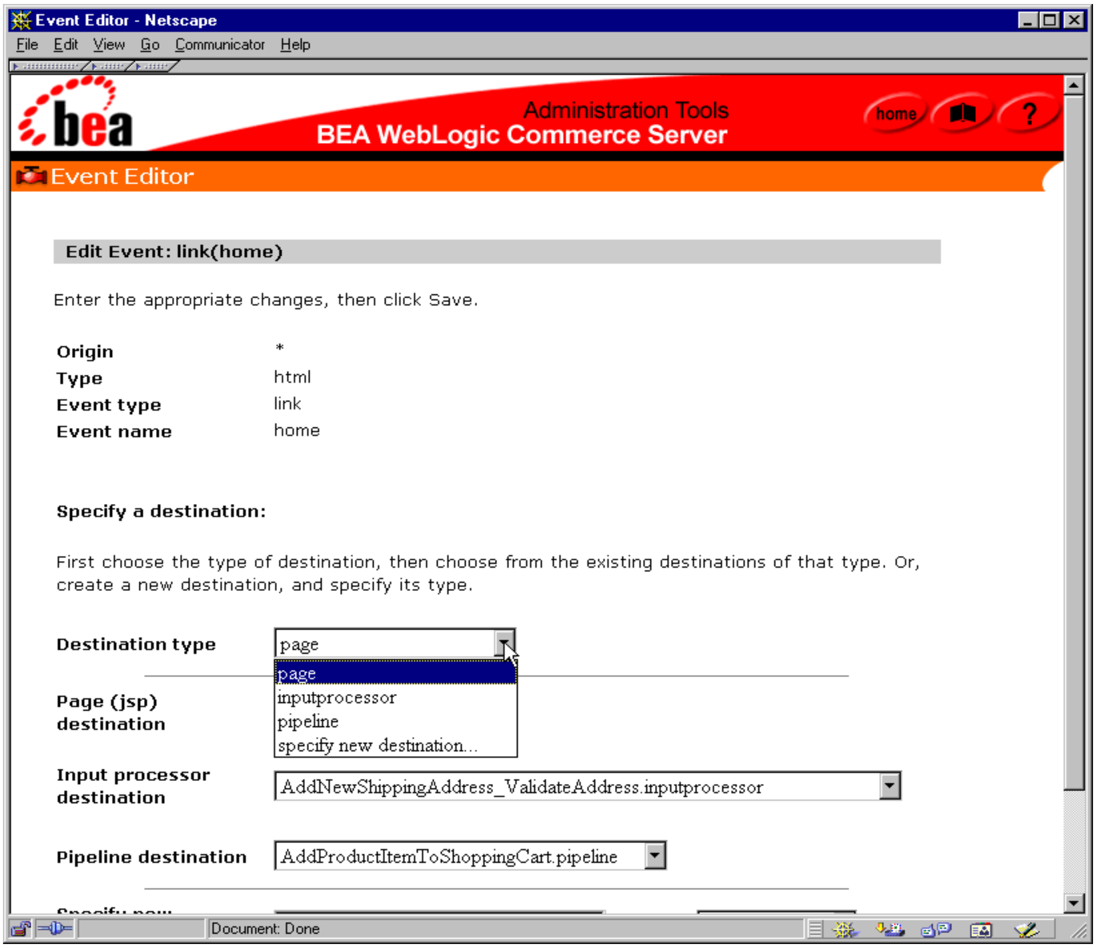
Editing an Event

To edit an event already associated with a global origin, follow these steps:

1. Verify that the global origin for which you want to edit an event is expanded (that is, an open arrow is shown to the left of the global origin, and the event you want to modify is shown). If the global origin is not expanded, click the solid arrow located to the left of the global origin to expand it.

2. Click a hyperlinked event to edit it (for example, click `link(home)` → `index.jsp`). The Edit Event page appears, containing the original values for each field, as shown in Figure 3-7.

Figure 3-7 Global Origin: Edit Event



Edit Event: link(home)

Enter the appropriate changes, then click Save.

Origin	*
Type	html
Event type	link
Event name	home

Specify a destination:

First choose the type of destination, then choose from the existing destinations of that type. Or, create a new destination, and specify its type.

Destination type	page
Page (jsp) destination	
Input processor destination	AddNewShippingAddress_ValidateAddress.inputprocessor
Pipeline destination	AddProductItemToShoppingCart.pipeline

Note: The Edit Event page does not allow you to modify the event type or event name. To do so, you must first delete the event using the instructions provided in “Deleting an Event” on page 3-14 and then recreate it using the instructions provided in “Adding an Event” on page 3-10.

3. Recall that an event destination could be an existing page (`.htm`, `.html`, `.jsp`), input processor, or Pipeline, or an entirely new page, input processor, or Pipeline. To change the destination for the event, follow these steps:

a. Select the destination type from the Destination Type pull-down menu (one of Page, Input Processor, Pipeline, or Specify New Destination).

Note: Selecting a destination type in this menu disables form fields that are not required for the chosen destination type. To enable unavailable form fields, choose a different destination type.

b. If you selected Page from the Destination Type pull-down menu, select an existing page (`.htm`, `.html`, or `.jsp`) from the Page (JSP) Destination pull-down menu. If you selected Input Processor, select an existing input processor from the Input Processor Destination pull-down menu. If you selected Pipeline, select an existing Pipeline from the Pipeline Destination pull-down menu. If you selected Specify New Destination from the Destination Type pull-down menu, enter the name of the destination in the Specify New Destination text field, and select the destination type from the Of Type pull-down menu (jsp, htm, html, inputprocessor, or pipeline).

c. Click the Save button to save the changes to the event. You are returned to the Global Origins page, and the modified event (marked by an open diamond) appears beneath the appropriate global origin in an alphabetic list.

Deleting an Event

To delete an event currently associated with a global origin, follow these steps:

1. Verify that the global origin for which you want to delete an event is expanded (that is, an open arrow is shown to the left of the global origin, and the event you want to delete is shown). If the global origin is not expanded, click the solid arrow located to the left of the global origin to expand it.
2. To delete the event, click the red X located to the right of the event.
3. Click the OK button in the confirmation pop-up window.

Note: If you delete all the events associated with a global origin and then leave the Global Origins page (or click the arrow next to the origin to collapse it), the origin itself will also be deleted. This is done to prevent error conditions in the Webflow. However, if you just delete all the events associated with a global origin and immediately start to add new events to the global origin, the origin itself will remain. This way, you do not always have to recreate the origin after deleting all of its events.

Deleting a Global Origin

To delete a global origin, click the red X located to the right of the origin. Confirm the deletion by clicking the OK button.

Warning: Deleting a global origin means that the origin and all of its associated events will be deleted.

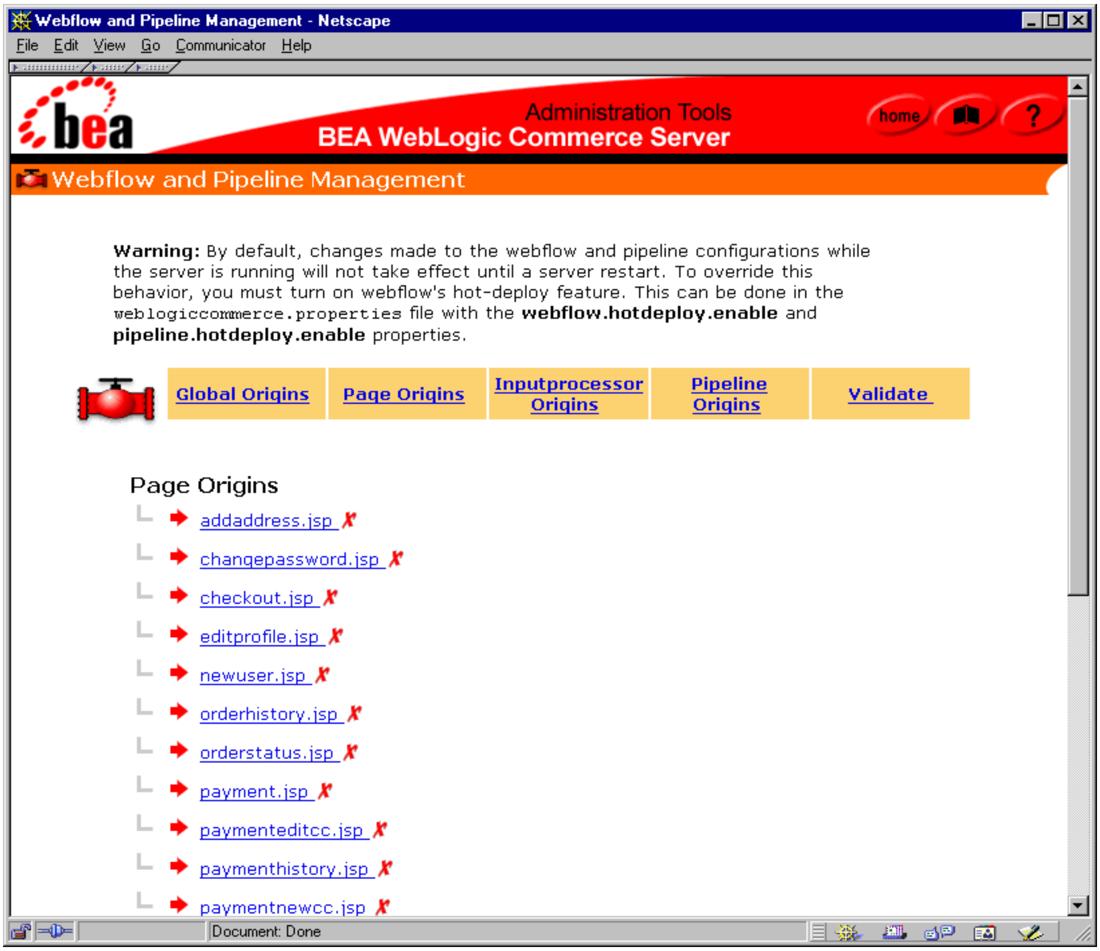
Page Origins

A page is any file with a `.htm`, `.html`, or `.jsp` extension. When a page appears to the left-hand side of the equal sign in the Webflow syntax, the page represents the current state and is considered an origin. Page origins require a result (destination) state to be defined for a link or button event, which is specified to the right-hand side of the equal sign in the Webflow syntax. Together, the complete line is referred to as a page Webflow transition.

Note: For more information about page origins and the Webflow syntax, see “Customizing Webflow Using the webflow.properties File” on page 2-2.

The Page Origins portion of the Webflow and Pipeline Editor allows you to provide all relevant information about the pages in your Web site. To work with page origins, click the Page Origins link (or its corresponding Edit button). The Page Origins page appears, as shown in Figure 3-8.

Figure 3-8 Page Origins


Webflow and Pipeline Management - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow and Pipeline Management

Warning: By default, changes made to the webflow and pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on webflow's hot-deploy feature. This can be done in the `weblogiccommerce.properties` file with the **webflow.hotdeploy.enable** and **pipeline.hotdeploy.enable** properties.

Global Origins
Page Origins
Inputprocessor Origins
Pipeline Origins
Validate

Page Origins

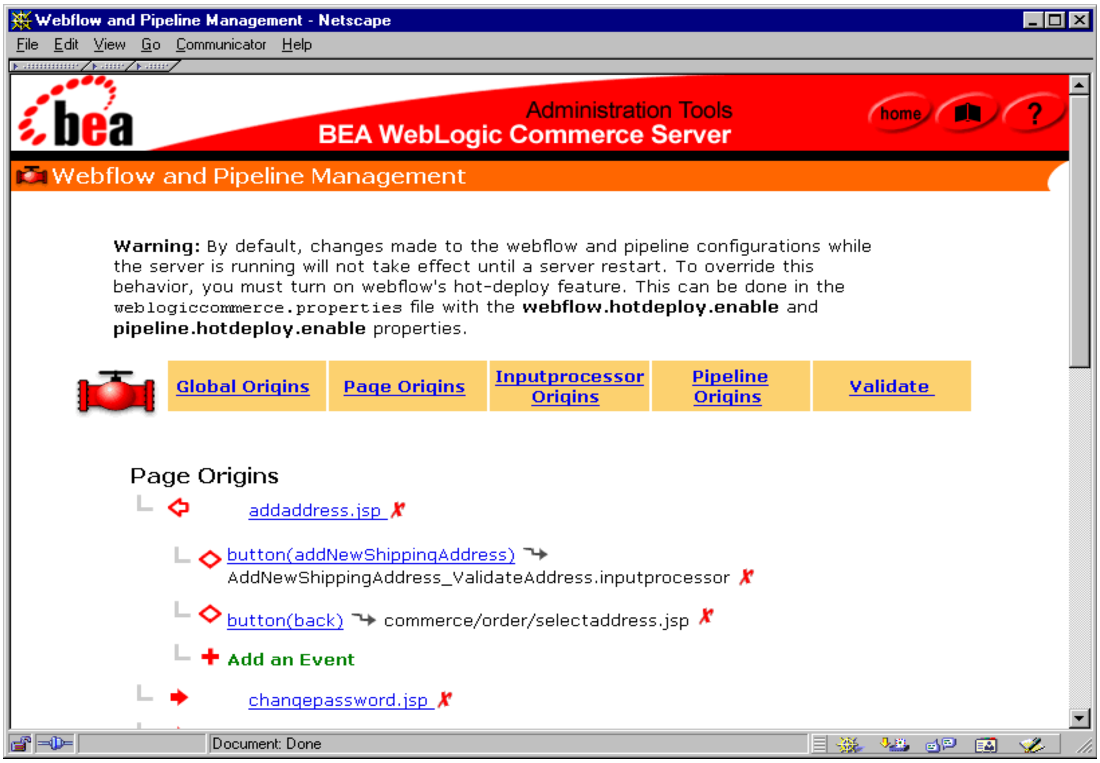
- addaddress.jsp ✗
- changepassword.jsp ✗
- checkout.jsp ✗
- editprofile.jsp ✗
- newuser.jsp ✗
- orderhistory.jsp ✗
- orderstatus.jsp ✗
- payment.jsp ✗
- paymenteditcc.jsp ✗
- paymenthistory.jsp ✗
- paymentnewcc.jsp ✗

From this page, you can view, add, and delete page origins. You can also edit page origins by modifying their name, type, and associated events, or by associating new events.

Viewing Page Origins

Before working with page origins, you may first want to see what page origins are currently in the Webflow. When the Page Origins page is first loaded into your browser, you will see a list of the current page origins, but no details about them. To view a page origin's associated events, click the solid arrow located to the left of the page origin. The arrow changes to an open arrow, and the events currently associated with the global origin appear with an open diamond, as shown in Figure 3-9.

Figure 3-9 Viewing Events for the addaddress.jsp Page Origin


Webflow and Pipeline Management - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow and Pipeline Management
Global Origins
Page Origins
Inputprocessor Origins
Pipeline Origins
Validate
Document: Done

Warning: By default, changes made to the webflow and pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on webflow's hot-deploy feature. This can be done in the `weblogiccommerce.properties` file with the **webflow.hotdeploy.enable** and **pipeline.hotdeploy.enable** properties.

Page Origins

addaddress.jsp

button(addNewShippingAddress) AddNewShippingAddress_ValidateAddress.inputprocessor

button(back) commerce/order/selectaddress.jsp

Add an Event

changepassword.jsp

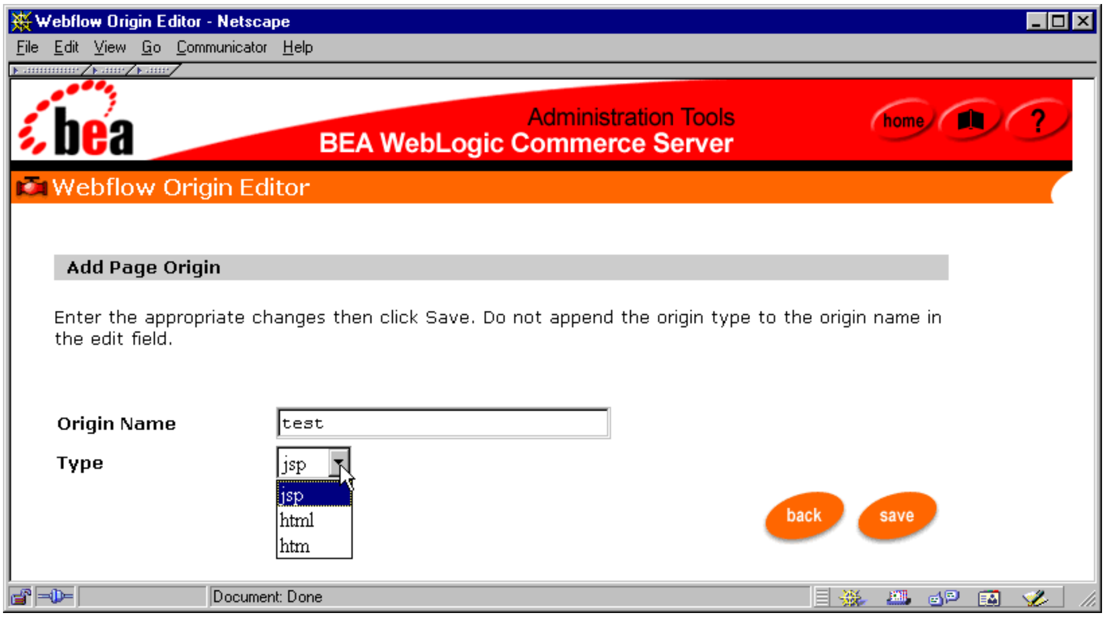
Note: The arrows to the left of each page origin are solid and point toward the origin if it can be expanded to show associated events. If the event information associated with a page origin can be collapsed, the arrow is open and points away from the origin. The `addaddress.jsp` page origin shown in Figure 3-9 can be collapsed. Events are always represented by open diamonds.

Adding a Page Origin

To add a page origin, follow these steps:

1. Click the plus (+) symbol located to the left of Add a Page Origin. (If the list of page origins is long, you may need to use the scrollbar.) The Add Page Origin page appears, as shown in Figure 3-10.

Figure 3-10 Add Page Origin


Webflow Origin Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow Origin Editor
back
save
Document: Done

Add Page Origin

Enter the appropriate changes then click Save. Do not append the origin type to the origin name in the edit field.

Origin Name: test

Type: jsp

jsp
html
htm

2. Enter the origin (page) name in the Origin Name text field. Do not include the file extension (type) in this field.

3. Select the type of page origin (`jsp`, `html`, `htm`) you want to add from the Type pull-down menu.

4. Click the Save button to save the new page origin. You are returned to the Page Origins page, and the new origin appears alphabetically in the list.

5. Click the solid arrow located to the right of the new page origin to view its associated events. The arrow changes to an open arrow pointing in the opposite direction. The default event of `link(home)` → `index.jsp` (marked by an open diamond) and an option to Add an Event are shown.

6. If you want to add a different event to the page origin, modify the default event, or delete the default event, proceed to the appropriate section in "Modifying a Page Origin's Events" on page 3-20.

Editing a Page Origin

You can edit a page origin in two ways:

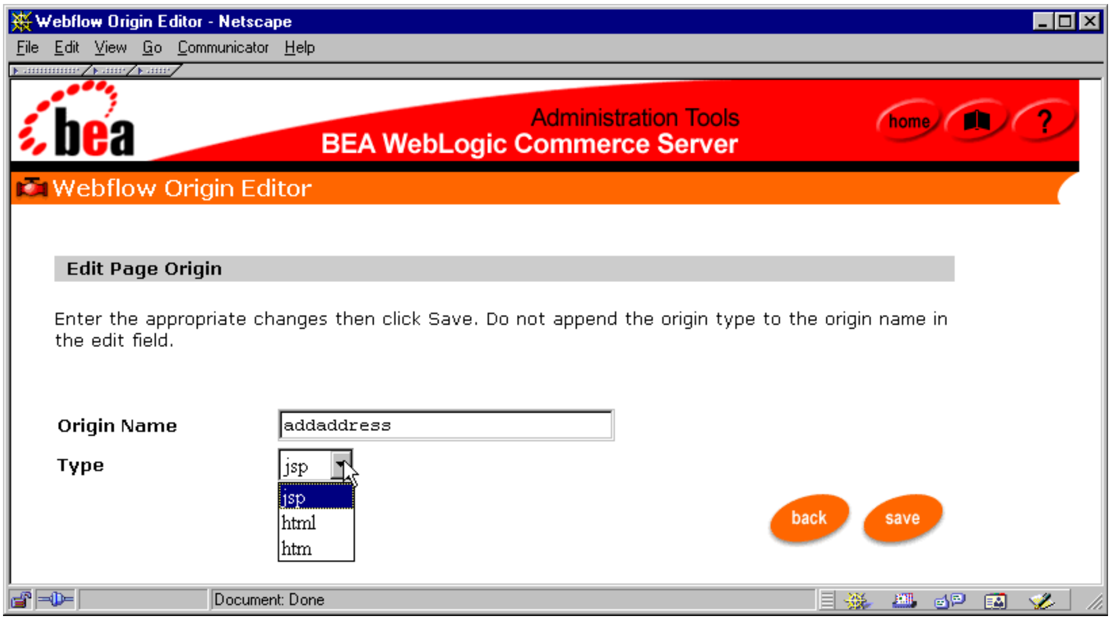
- By modifying information about the page origin itself (that is, its name or file extension/type).
- By modifying the page origin's associated events.

Modifying Information About a Page Origin

To modify information about a page origin, follow these steps:

1. Click a hyperlinked page origin to edit it (for example, click `addaddress.jsp`). The Edit Page Origin page appears, containing the original values for each field, as shown in Figure 3-11.

Figure 3-11 Edit Page Origin


Webflow Origin Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow Origin Editor
back
save
Document: Done

Edit Page Origin

Enter the appropriate changes then click Save. Do not append the origin type to the origin name in the edit field.

Origin Name	addaddress
Type	jsp (jsp, html, htm)

2. Change the origin (page) name or type of page origin using the form fields.

3. Click the Save button to save your changes. You are returned to the Page Origins page, and the modified page origin is shown alphabetically in the list.

Note: Modifications made to the page origin name and type could potentially affect other areas of the Webflow that utilize it. Be sure your modifications take this into account, and be sure to verify your modified properties file with the validation tool. For more information about the validation tool, see “How to Validate Your Properties Files” on page 3-58.

Modifying a Page Origin’s Events

You can also edit a page origin by modifying its associated events. You can add, edit, or delete events associated with a page origin as described in the following sections.

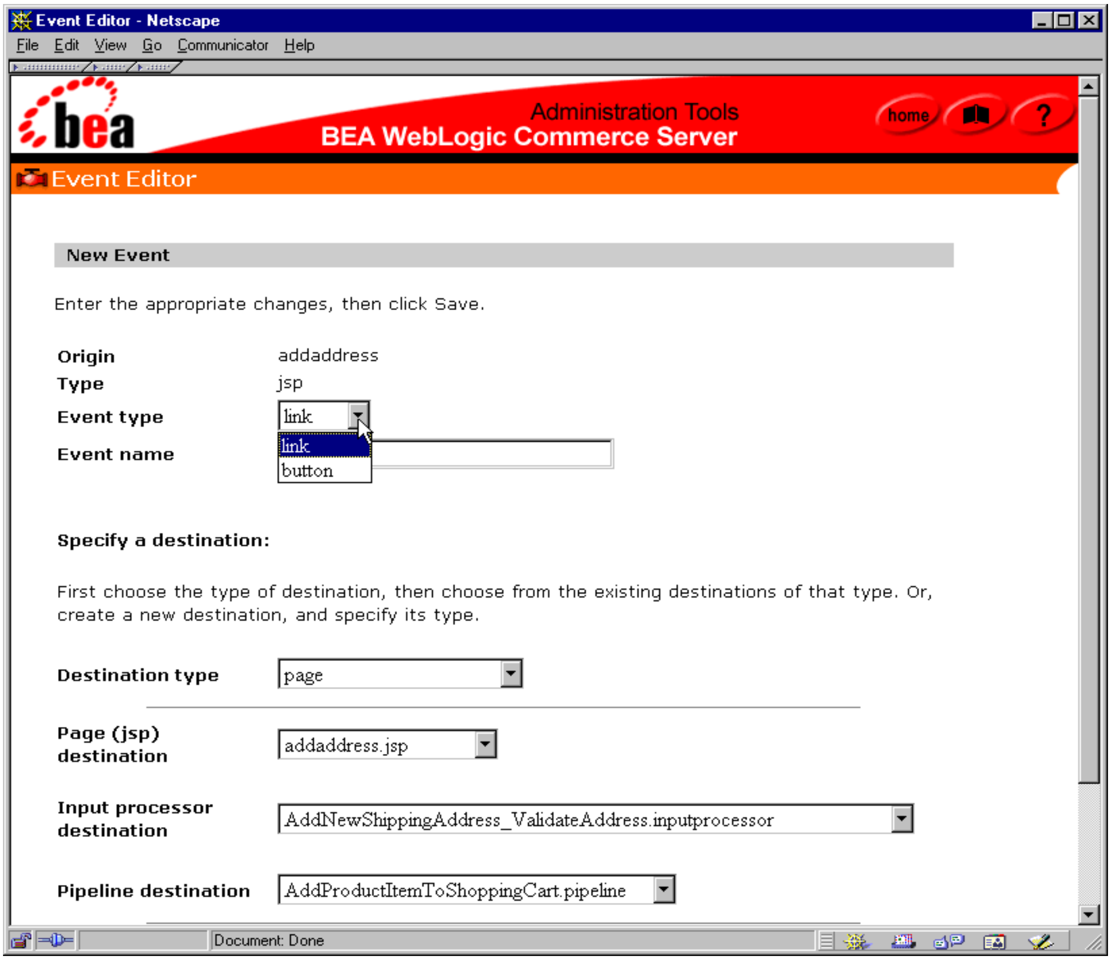
Adding an Event

To add an event to a page origin, follow these steps:

1. Verify that the page origin to which you want to add the event is expanded (that is, an open arrow is shown to the left of the page origin, and the Add an Event option is shown). If the page origin is not expanded, click the solid arrow located to the left of the page origin to expand it.

2. Click the plus (+) sign located next to Add an Event. (If the list of events is long, you may need to use the scrollbar.) The New Event page appears, as shown in Figure 3-12.

Figure 3-12 Page Origin: New Event



New Event

Enter the appropriate changes, then click Save.

Origin	addaddress
Type	jsp
Event type	link (link, button)
Event name	

Specify a destination:

First choose the type of destination, then choose from the existing destinations of that type. Or, create a new destination, and specify its type.

Destination type	page
Page (jsp) destination	addaddress.jsp
Input processor destination	AddNewShippingAddress_ValidateAddress.inputprocessor
Pipeline destination	AddProductItemToShoppingCart.pipeline

3. Select the appropriate Event Type from the pull-down menu, and enter a name for the event in the Event Name text field.

Note: For more information about event types and event names, see "Syntax of the webflow.properties File" on page 2-3 and "About Event Names" on page 2-4, respectively.

4. Next, you need to specify a destination for the event. The destination could be an existing page (`.htm`, `.html`, or `.jsp`), input processor, or Pipeline, or the destination could be an entirely new page, input processor, or Pipeline. To specify the destination, follow these steps:

   a. Select the destination type from the Destination Type pull-down menu (one of Page, Input Processor, Pipeline, or Specify New Destination).

Note: Selecting a destination type in this menu disables form fields that are not required for the chosen destination type. To enable unavailable form fields, choose a different destination type.

   b. If you selected Page from the Destination Type pull-down menu, select an existing page (`.htm`, `.html`, or `.jsp`) from the Page (JSP) Destination pull-down menu. If you selected Input Processor, select an existing input processor from the Input Processor Destination pull-down menu. If you selected Pipeline, select an existing Pipeline from the Pipeline Destination pull-down menu. If you selected Specify New Destination from the Destination Type pull-down menu, enter the name of the destination in the Specify New Destination text field, and select the destination type from the Of Type pull-down menu (jsp, htm, html, inputprocessor, or pipeline).

   c. Click the Save button to save the new event. You are returned to the Page Origins page, and the new event (marked by an open diamond) appears beneath the appropriate page origin in an alphabetic list.

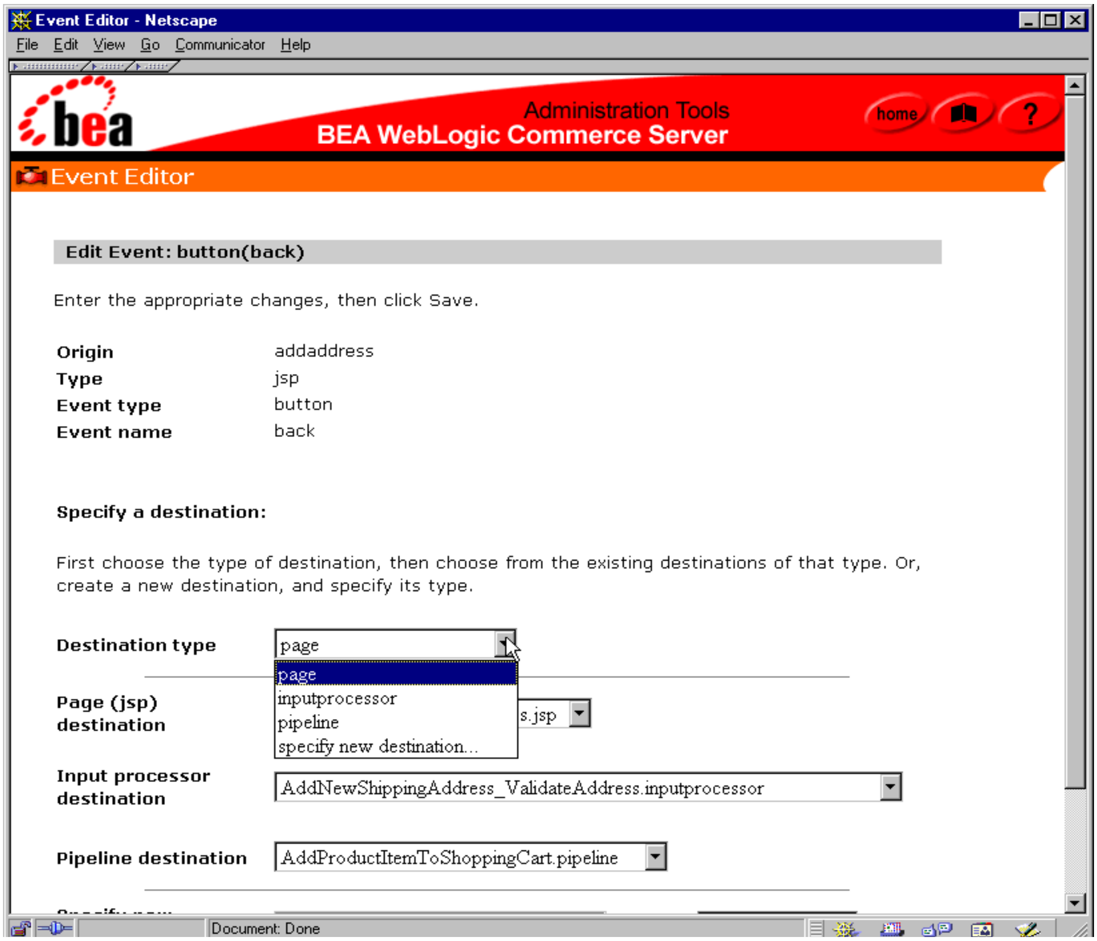
Editing an Event

To edit an event already associated with a page origin, follow these steps:

1. Verify that the page origin for which you want to edit an event is expanded (that is, an open arrow is shown to the left of the page origin, and the event you want to modify is shown). If the page origin is not expanded, click the solid arrow located to the left of the page origin to expand it.

2. Click a hyperlinked event to edit it (for example, click `button(back)` → `commerce/order/selectaddress.jsp`). The Edit Event page appears, containing the original values for each field, as shown in Figure 3-13.

Figure 3-13 Page Origin: Edit Event



Edit Event: button(back)

Enter the appropriate changes, then click Save.

Origin	addaddress
Type	jsp
Event type	button
Event name	back

Specify a destination:

First choose the type of destination, then choose from the existing destinations of that type. Or, create a new destination, and specify its type.

Destination type	page (page / inputprocessor / pipeline / specify new destination...)
Page (jsp) destination	s.jsp
Input processor destination	AddNewShippingAddress_ValidateAddress.inputprocessor
Pipeline destination	AddProductItemToShoppingCart.pipeline

Note: The Edit Event page does not allow you to modify the event type or event name. To do so, you must first delete the event using the instructions provided in “Deleting an Event” on page 3-24 and then recreate it using the instructions provided in “Adding an Event” on page 3-20.

3. Recall that an event destination could be an existing page (`.htm`, `.html`, `.jsp`), input processor, or Pipeline, or an entirely new page, input processor, or Pipeline. To change the destination for the event, follow these steps:

   a. Select the destination type from the Destination Type pull-down menu (one of Page, Input Processor, Pipeline, or Specify New Destination).

Note: Selecting a destination type in this menu disables form fields that are not required for the chosen destination type. To enable unavailable form fields, choose a different destination type.

   b. If you selected Page from the Destination Type pull-down menu, select an existing page (`.htm`, `.html`, or `.jsp`) from the Page (JSP) Destination pull-down menu. If you selected Input Processor, select an existing input processor from the Input Processor Destination pull-down menu. If you selected Pipeline, select an existing Pipeline from the Pipeline Destination pull-down menu. If you selected Specify New Destination from the Destination Type pull-down menu, enter the name of the destination in the Specify New Destination text field, and select the destination type from the Of Type pull-down menu (jsp, htm, html, inputprocessor, or pipeline).

   c. Click the Save button to save the changes to the event. You are returned to the Page Origins page, and the modified event (marked by an open diamond) appears beneath the appropriate page origin in an alphabetic list.

Deleting an Event

To delete an event currently associated with a page origin, follow these steps:

1. Verify that the page origin for which you want to delete an event is expanded (that is, an open arrow is shown to the left of the page origin, and the event you want to delete is shown). If the page origin is not expanded, click the solid arrow located to the left of the page origin to expand it.

2. To delete the event, click the red X located to the right of the event.

3. Click the OK button in the confirmation pop-up window.

Note: If you delete all the events associated with a page origin and then leave the Page Origins page (or click the arrow next to the origin to collapse it), the origin itself will also be deleted. This is done to prevent error conditions in the Webflow. However, if you just delete all the events associated with a page origin and immediately start to add new events to the page origin, the origin itself will remain. This way, you do not always have to recreate the origin after deleting all of its events.

Deleting a Page Origin

To delete a page origin, click the red X located to the right of the origin. Confirm the deletion by clicking the OK button.

Warning: Deleting a page origin means that the origin and all of its associated events will be deleted.

Inputprocessor Origins

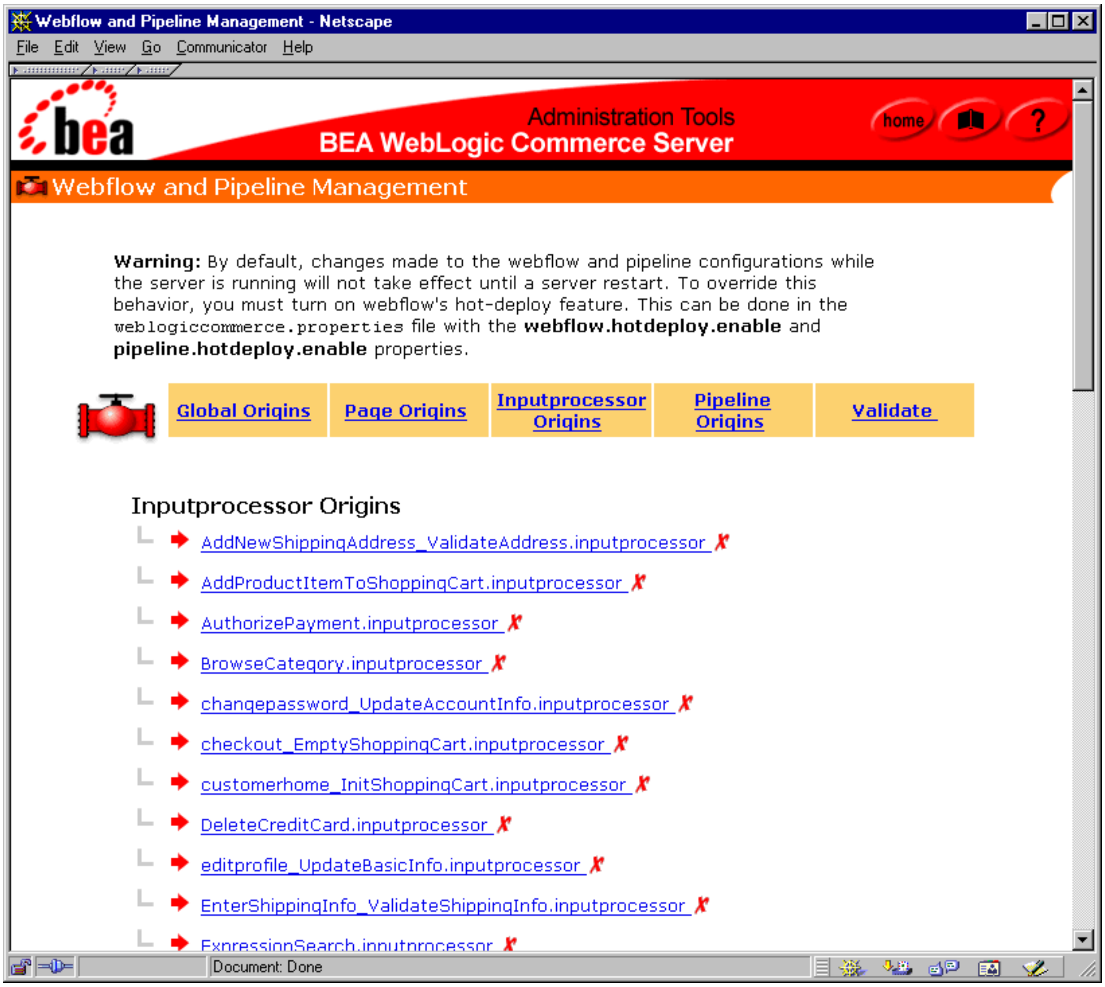
Input processors are classes that validate customer-supplied information within a form. When a customer submits a form, an input processor decides whether to reload the form and point out errors, or allow the customer to continue. Therefore, an input processor often appears to the left-hand side of the equal sign in the Webflow syntax. When this is the case, the input processor represents the current state and is considered an origin. Input processor origins require a result (destination) state to be defined for the success event, which is specified to the right-hand side of the equal sign. Together, the complete line is referred to as an input processor transition.

Notes: It is suggested that you also provide a result state for the exception event.

For more information about input processor origins and Webflow syntax, see “Syntax of Input Processors in the webflow.properties File” on page 2-10.

The Inputprocessor Origins portion of the Webflow and Pipeline Editor allows you to provide all relevant information about your input processors. To work with input processor origins, click the Inputprocessor Origins link (or its corresponding Edit button). The Inputprocessor Origins page appears, as shown in Figure 3-14.

Figure 3-14 Inputprocessor Origins


Webflow and Pipeline Management - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow and Pipeline Management

Warning: By default, changes made to the webflow and pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on webflow's hot-deploy feature. This can be done in the `weblogiccommerce.properties` file with the **webflow.hotdeploy.enable** and **pipeline.hotdeploy.enable** properties.

Global Origins
Page Origins
Inputprocessor Origins
Pipeline Origins
Validate

Inputprocessor Origins

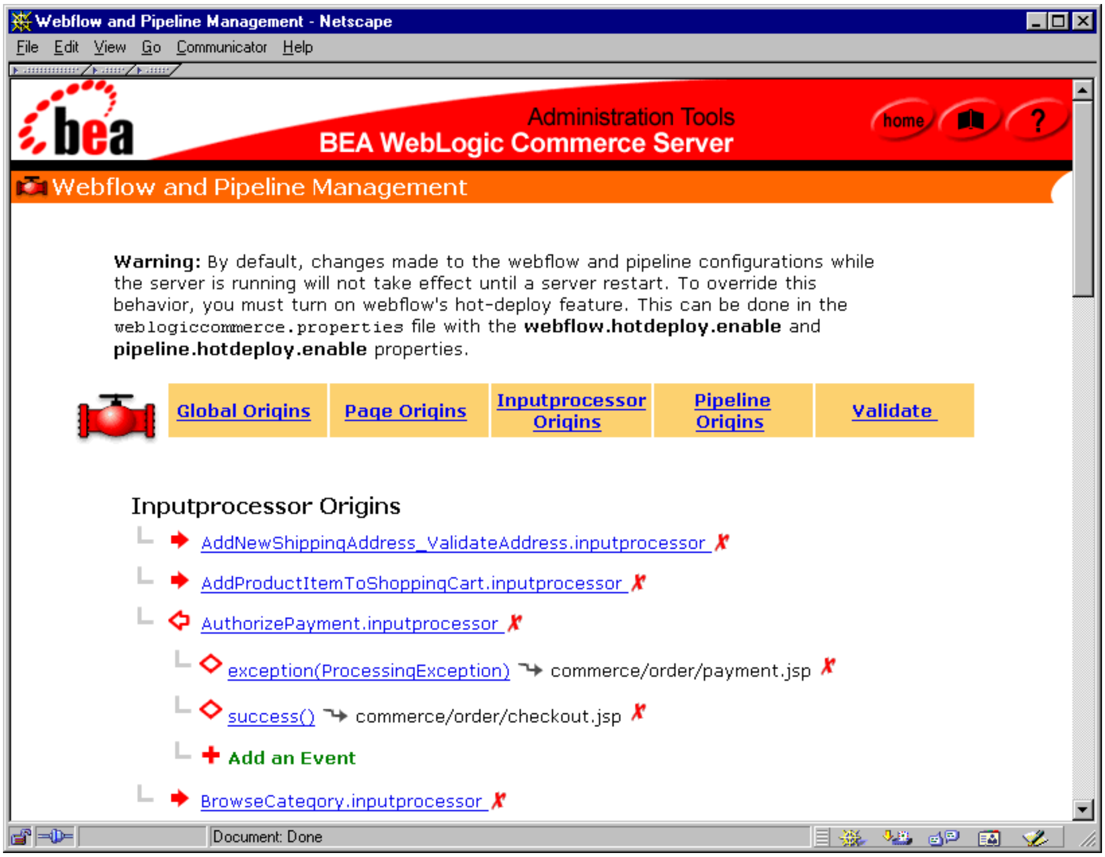
- AddNewShippingAddress_ValidateAddress.inputprocessor
- AddProductItemToShoppingCart.inputprocessor
- AuthorizePayment.inputprocessor
- BrowseCategory.inputprocessor
- changepassword_UpdateAccountInfo.inputprocessor
- checkout_EmptyShoppingCart.inputprocessor
- customerhome_InitShoppingCart.inputprocessor
- DeleteCreditCard.inputprocessor
- editprofile_UpdateBasicInfo.inputprocessor
- EnterShippingInfo_ValidateShippingInfo.inputprocessor
- ExpressionSearch.inputprocessor

From this page, you can view, add, and delete input processor origins. You can also edit input processor origins by modifying their name, class name, and associated events, or by associating new events.

Viewing Inputprocessor Origins

Before working with input processor origins, you may first want to see what input processor origins are currently in the Webflow. When the Inputprocessor Origins page is first loaded into your browser, you will see a list of the current input processor origins, but no details about them. To view an input processor origin's associated events, click the solid arrow located to the left of the input processor origin. The arrow changes to an open arrow, and the events currently associated with the input processor origin appear with an open diamond, as shown in Figure 3-15.

Figure 3-15 Viewing Events for the AuthorizePayment.inputprocessor Origin


Webflow and Pipeline Management - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow and Pipeline Management
Global Origins
Page Origins
Inputprocessor Origins
Pipeline Origins
Validate
Document: Done

Warning: By default, changes made to the webflow and pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on webflow's hot-deploy feature. This can be done in the `weblogiccommerce.properties` file with the **webflow.hotdeploy.enable** and **pipeline.hotdeploy.enable** properties.

Inputprocessor Origins

- AddNewShippingAddress_ValidateAddress.inputprocessor
- AddProductItemToShoppingCart.inputprocessor
- AuthorizePayment.inputprocessor
  - exception(ProcessingException) → commerce/order/payment.jsp
  - success() → commerce/order/checkout.jsp
  - Add an Event
- BrowseCategory.inputprocessor

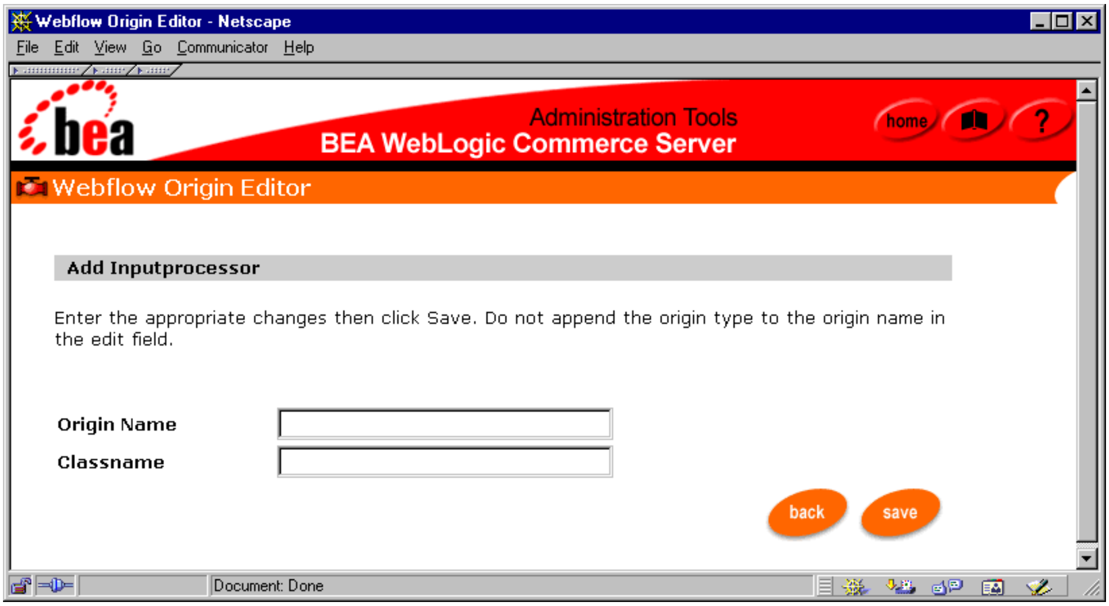
Note: The arrows to the left of each input processor origin are solid and point toward the origin if it can be expanded to show associated events. If the event information associated with an input processor origin can be collapsed, the arrow is open and points away from the origin. The `AddProductItemToShoppingCart.inputprocessor` input processor origin in Figure 3-15 is an example of an origin that can be expanded; the `AuthorizePayment.inputprocessor` input processor origin is an example of an origin that can be collapsed. Events are always represented by open diamonds.

Adding an Inputprocessor Origin

To add an input processor origin, follow these steps:

1. Click the plus (+) symbol located to the left of Add an Inputprocessor Origin. (If the list of input processor origins is long, you may need to use the scrollbar.) The Add Inputprocessor Origin page appears, as shown in Figure 3-16.

Figure 3-16 Add Inputprocessor Origin


Webflow Origin Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow Origin Editor

Add Inputprocessor

Enter the appropriate changes then click Save. Do not append the origin type to the origin name in the edit field.

Origin Name	
Classname	

2. Enter the origin (input processor) name in the Origin Name text field.
3. Enter the full class name for the input processor in the Classname text field.
4. Click the Save button to save the new input processor origin. You are returned to the Inputprocessor Origins page, and the new origin appears alphabetically in the list.
5. Click the solid arrow located to the right of the new input processor origin to view its associated events. The arrow changes to an open arrow pointing in the opposite direction. The default event of `success()` → `index.jsp` (marked by an open diamond) and an option to Add an Event are shown.
6. If you want to add a different event to the input processor origin, modify the default event, or delete the default event, proceed to the appropriate section in "Modifying an Inputprocessor Origin's Events" on page 3-30.

Editing an Inputprocessor Origin

You can edit an input processor origin in two ways:

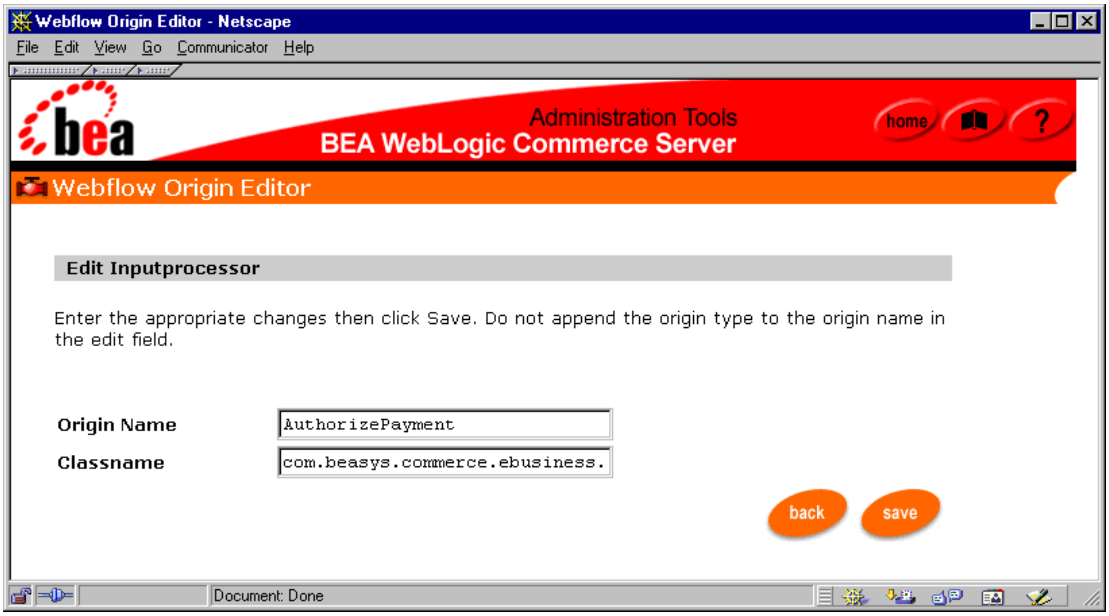
- By modifying information about the input processor origin itself (that is, its name or class name).
- By modifying the input processor origin's associated events.

Modifying Information About an Inputprocessor Origin

To modify information about an input processor origin, follow these steps:

1. Click a hyperlinked input processor origin to edit it (for example, click `AuthorizePayment.inputprocessor`). The Edit Inputprocessor Origin page appears, containing the original values for each field, as shown in Figure 3-17.

Figure 3-17 Edit Inputprocessor Origin


Webflow Origin Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow Origin Editor
back
save
Document: Done

Edit Inputprocessor

Enter the appropriate changes then click Save. Do not append the origin type to the origin name in the edit field.

Origin Name	AuthorizePayment
Classname	com.beasys.commerce.ebusiness.

2. Change the origin (input processor) name or class name using the form fields.

3. Click the Save button to save your changes. You are returned to the Inputprocessor Origins page, and the modified input processor origin is shown alphabetically in the list.

Note: Modifications made to the input processor origin name and class name could potentially affect other areas of the Webflow that utilize it. Be sure your modifications take this into account, and be sure to verify your modified properties file with the validation tool. For more information about the validation tool, see “How to Validate Your Properties Files” on page 3-58.

Modifying an Inputprocessor Origin’s Events

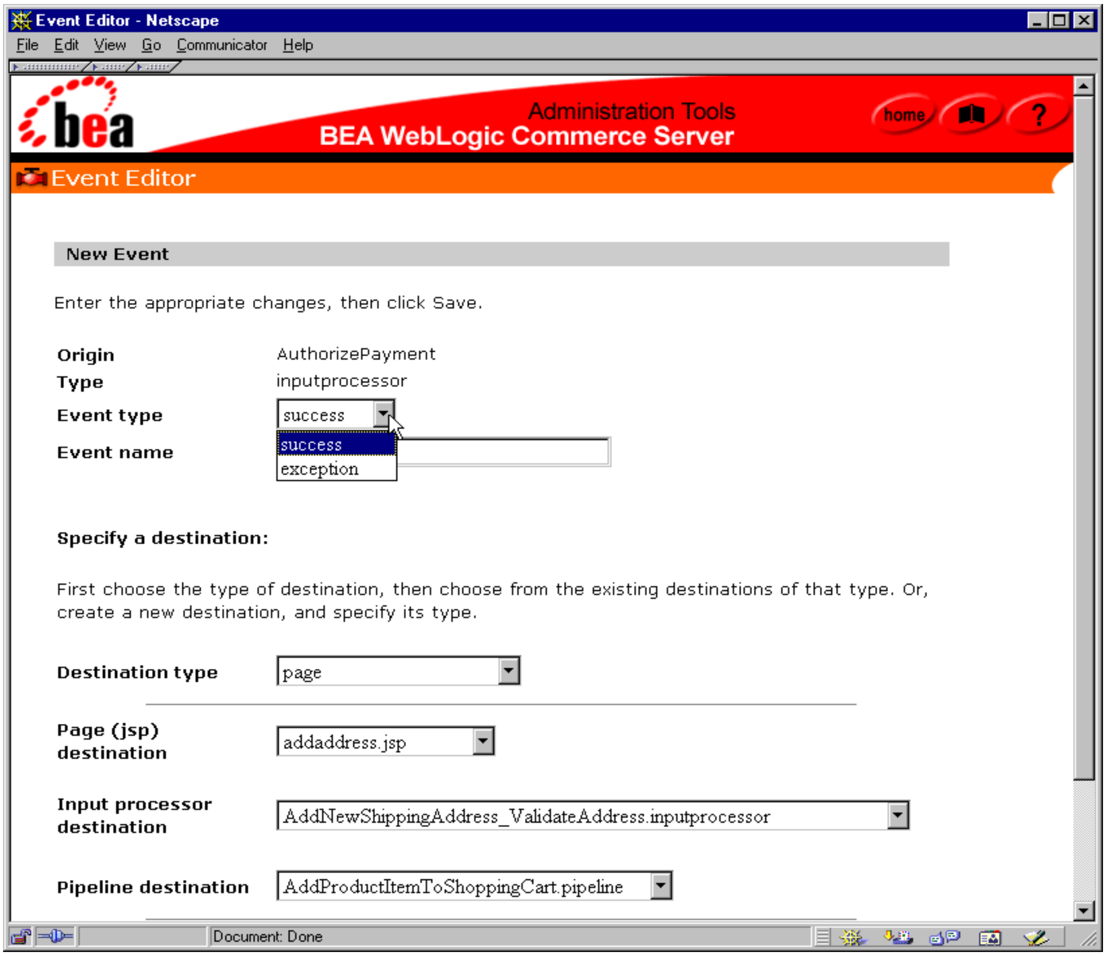
You can also edit an input processor origin by modifying its associated events. You can add, edit, or delete events associated with an input processor origin as described in the following sections.

Adding an Event

To add an event to an input processor origin, follow these steps:

1. Verify that the input processor origin to which you want to add the event is expanded (that is, an open arrow is shown to the left of the input processor origin, and the Add an Event option is shown). If the input processor origin is not expanded, click the solid arrow located to the left of the input processor origin to expand it.
2. Click the plus (+) sign located next to Add an Event. (If the list of events is long, you may need to use the scrollbar.) The New Event page appears, as shown in Figure 3-18.

Figure 3-18 Inputprocessor Origin: New Event


Event Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Event Editor

New Event

Enter the appropriate changes, then click Save.

Origin	AuthorizePayment
Type	inputprocessor
Event type	success (options: success, exception)
Event name	

Specify a destination:

First choose the type of destination, then choose from the existing destinations of that type. Or, create a new destination, and specify its type.

Destination type	page
Page (jsp) destination	addaddress.jsp
Input processor destination	AddNewShippingAddress_ValidateAddress.inputprocessor
Pipeline destination	AddProductItemToShoppingCart.pipeline

3. Select the appropriate Event Type from the pull-down menu, and enter a name for the event in the Event Name text field.

Note: For more information about event types and event names, see "Syntax of the webflow.properties File" on page 2-3 and "About Event Names" on page 2-4, respectively.

4. Next, you need to specify a destination for the event. The destination could be an existing page (`.htm`, `.html`, or `.jsp`), input processor, or Pipeline, or the destination could be an entirely new page, input processor, or Pipeline. To specify the destination, follow these steps:

   a. Select the destination type from the Destination Type pull-down menu (one of Page, Input Processor, Pipeline, or Specify New Destination).

Note: Selecting a destination type in this menu disables form fields that are not required for the chosen destination type. To enable unavailable form fields, choose a different destination type.

   b. If you selected Page from the Destination Type pull-down menu, select an existing page (`.htm`, `.html`, or `.jsp`) from the Page (JSP) Destination pull-down menu. If you selected Input Processor, select an existing input processor from the Input Processor Destination pull-down menu. If you selected Pipeline, select an existing Pipeline from the Pipeline Destination pull-down menu. If you selected Specify New Destination from the Destination Type pull-down menu, enter the name of the destination in the Specify New Destination text field, and select the destination type from the Of Type pull-down menu (jsp, htm, html, inputprocessor, or pipeline).

   c. Click the Save button to save the new event. You are returned to the Inputprocessor Origins page, and the new event (marked by an open diamond) appears beneath the appropriate input processor origin in an alphabetic list.

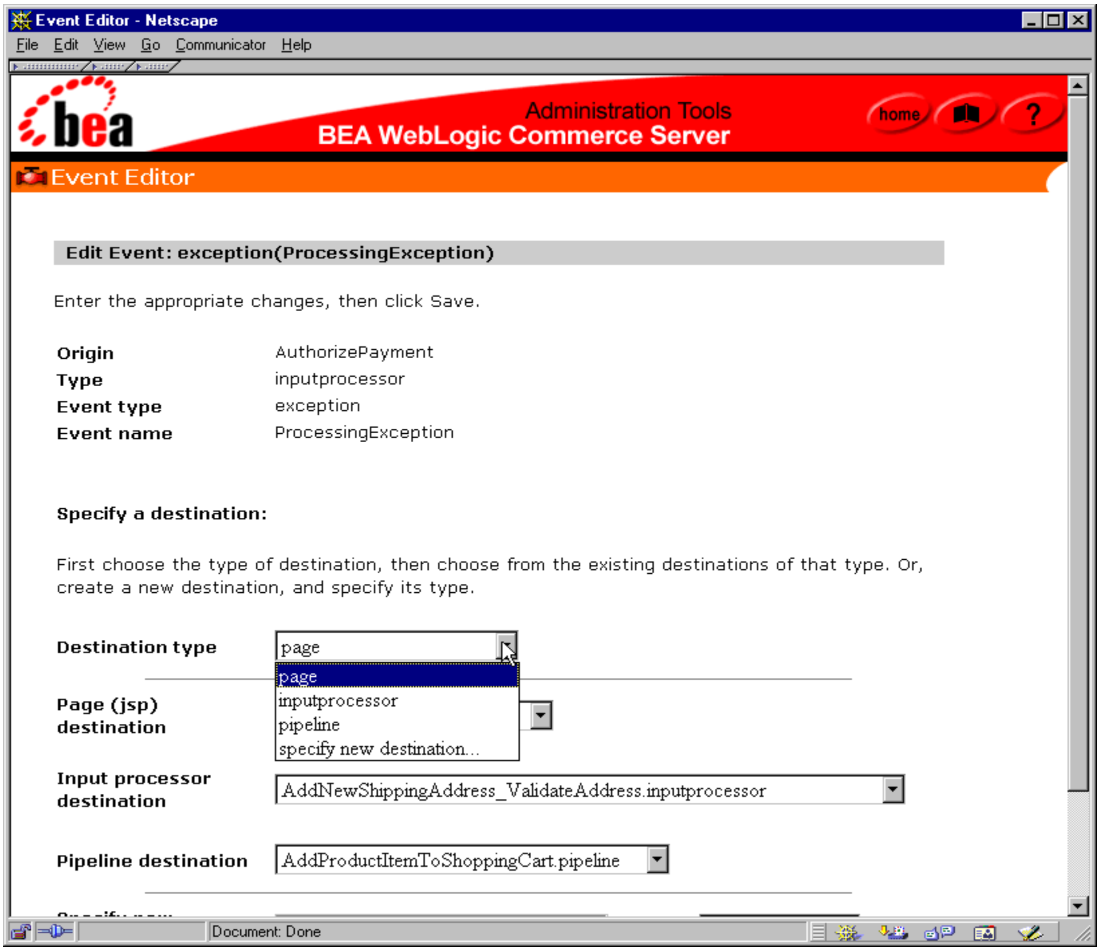
Editing an Event

To edit an event already associated with an input processor origin, follow these steps:

1. Verify that the input processor origin for which you want to edit an event is expanded (that is, an open arrow is shown to the left of the input processor origin, and the event you want to modify is shown). If the input processor origin is not expanded, click the solid arrow located to the left of the input processor origin to expand it.

2. Click a hyperlinked event to edit it (for example, click `exception(ProcessingException)` → `commerce/order/payment.jsp`). The Edit Event page appears, containing the original values for each field, as shown in Figure 3-19.

Figure 3-19 Inputprocessor Origin: Edit Event



Edit Event: exception(ProcessingException)

Enter the appropriate changes, then click Save.

Origin	AuthorizePayment
Type	inputprocessor
Event type	exception
Event name	ProcessingException

Specify a destination:

First choose the type of destination, then choose from the existing destinations of that type. Or, create a new destination, and specify its type.

Destination type	page (options: page, inputprocessor, pipeline, specify new destination...)
Page (jsp) destination	
Input processor destination	AddNewShippingAddress_ValidateAddress.inputprocessor
Pipeline destination	AddProductItemToShoppingCart.pipeline

Note: The Edit Event page does not allow you to modify the event type or event name. To do so, you must first delete the event using the instructions provided in “Deleting an Event” on page 3-35 and then recreate it using the instructions provided in “Adding an Event” on page 3-31.

3. Recall that an event destination could be an existing page (`.htm`, `.html`, `.jsp`), input processor, or Pipeline, or an entirely new page, input processor, or Pipeline. To change the destination for the event, follow these steps:

a. Select the destination type from the Destination Type pull-down menu (one of Page, Input Processor, Pipeline, or Specify New Destination).

Note: Selecting a destination type in this menu disables form fields that are not required for the chosen destination type. To enable unavailable form fields, choose a different destination type.

b. If you selected Page from the Destination Type pull-down menu, select an existing page (`.htm`, `.html`, or `.jsp`) from the Page (JSP) Destination pull-down menu. If you selected Input Processor, select an existing input processor from the Input Processor Destination pull-down menu. If you selected Pipeline, select an existing Pipeline from the Pipeline Destination pull-down menu. If you selected Specify New Destination from the Destination Type pull-down menu, enter the name of the destination in the Specify New Destination text field, and select the destination type from the Of Type pull-down menu (jsp, htm, html, inputprocessor, or pipeline).

c. Click the Save button to save the changes to the event. You are returned to the Inputprocessor Origins page, and the modified event (marked by an open diamond) appears beneath the appropriate input processor origin in an alphabetic list.

Deleting an Event

To delete an event currently associated with an input processor origin, follow these steps:

1. Verify that the input processor origin for which you want to delete an event is expanded (that is, an open arrow is shown to the left of the input processor origin, and the event you want to delete is shown). If the input processor origin is not expanded, click the solid arrow located to the left of the input processor origin to expand it.
2. To delete the event, click the red X located to the right of the event.
3. Click the OK button in the confirmation pop-up window.

Note: Unlike the other origins (global, page, and Pipeline), if you delete all the events associated with an input processor origin and then leave the Inputprocessor Origins page (or click the arrow next to the origin to collapse it), the origin itself will not be deleted. This is because an input processor origin with no events will not cause an error condition in the Webflow, as the other origins would.

Deleting an Inputprocessor Origin

To delete an input processor origin, click the red X located to the right of the origin. Confirm the deletion by clicking the OK button.

Warning: Deleting an input processor origin means that the origin and all of its associated events will be deleted.

Pipeline Origins

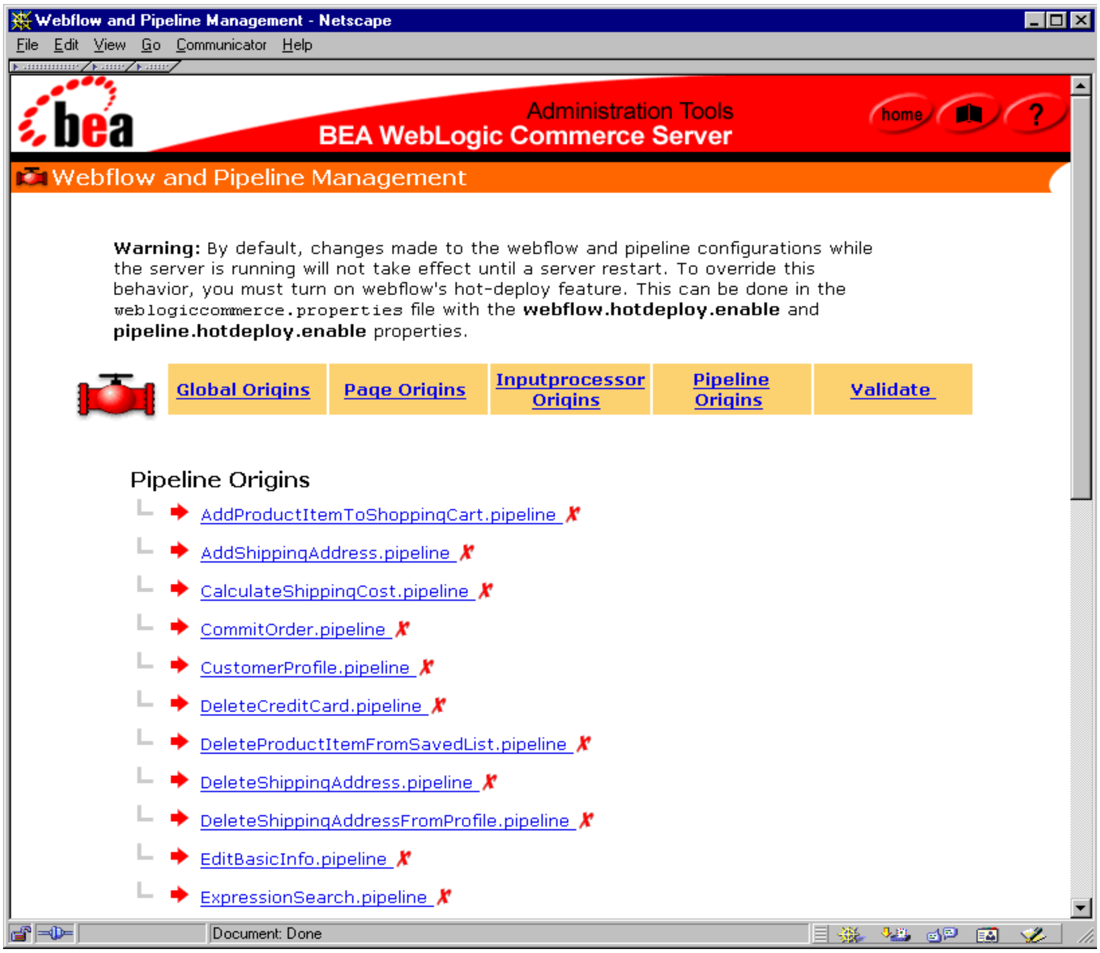
Pipelines are classes that handle the business logic or back-end processes of an e-business Web site. Therefore, a Pipeline often appears to the left-hand side of the equal sign in the Webflow syntax. When this is the case, the Pipeline represents the current state and is considered an origin. Pipeline origins require a result (destination) state to be defined for the success event, which is specified to the right-hand side of the equal sign in the Webflow syntax. Together, the complete line is referred to as a Pipeline transition.

Notes: It is suggested that you also provide a result state for the exception event.

For more information about Pipeline origins and the Webflow syntax see “Using Pipelines in the Webflow” on page 2-20.

The Pipeline Origins portion of the Webflow and Pipeline Editor allows you to provide all relevant information about your Pipelines. To work with Pipeline origins, click the Pipeline Origins link (or its corresponding Edit button). The Pipeline Origins page appears, as shown in Figure 3-20.

Figure 3-20 Pipeline Origins



Webflow and Pipeline Management - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow and Pipeline Management

Warning: By default, changes made to the webflow and pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on webflow's hot-deploy feature. This can be done in the `weblogiccommerce.properties` file with the **webflow.hotdeploy.enable** and **pipeline.hotdeploy.enable** properties.

Global Origins
Page Origins
Inputprocessor Origins
Pipeline Origins
Validate

Pipeline Origins

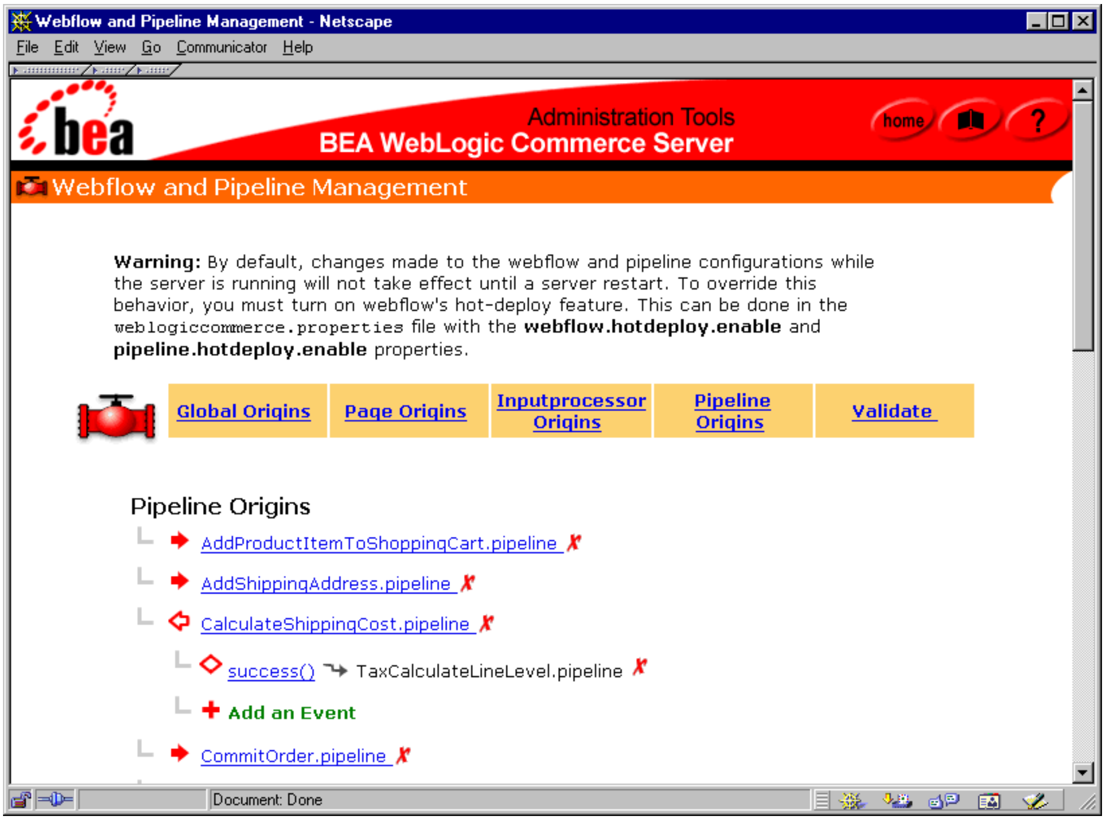
- AddProductItemToShoppingCart.pipeline
- AddShippingAddress.pipeline
- CalculateShippingCost.pipeline
- CommitOrder.pipeline
- CustomerProfile.pipeline
- DeleteCreditCard.pipeline
- DeleteProductItemFromSavedList.pipeline
- DeleteShippingAddress.pipeline
- DeleteShippingAddressFromProfile.pipeline
- EditBasicInfo.pipeline
- ExpressionSearch.pipeline

From this page, you can view, add, and delete Pipeline origins. You can also edit Pipeline origins by modifying their name or class name, working with the Pipeline's associated Pipeline components, or by modifying the Pipeline origin's associated events.

Viewing Pipeline Origins

Before working with Pipeline origins, you may first want to see what Pipeline origins are currently in the Webflow. When the Pipeline Origins page is first loaded into your browser, you will see a list of the current Pipeline origins, but no details about them. To view an Pipeline origin's associated events, click the solid arrow located to the left of the Pipeline origin. The arrow changes to an open arrow, and the events currently associated with the Pipeline origin appear with an open diamond, as shown in Figure 3-21.

Figure 3-21 Viewing Events for the AuthorizePayment.inputprocessor Origin


Webflow and Pipeline Management - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow and Pipeline Management

Warning: By default, changes made to the webflow and pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on webflow's hot-deploy feature. This can be done in the `weblogiccommerce.properties` file with the **webflow.hotdeploy.enable** and **pipeline.hotdeploy.enable** properties.

Global Origins
Page Origins
Inputprocessor Origins
Pipeline Origins
Validate

Pipeline Origins

- AddProductItemToShoppingCart.pipeline
- AddShippingAddress.pipeline
- CalculateShippingCost.pipeline
  - success() TaxCalculateLineLevel.pipeline
  - Add an Event
- CommitOrder.pipeline

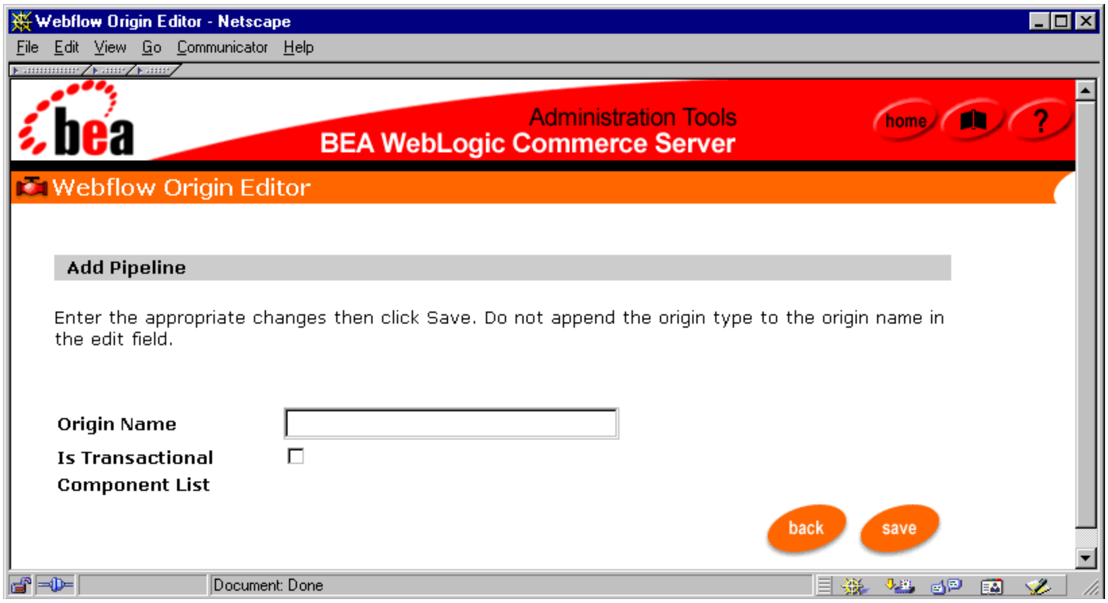
Note: The arrows to the left of each Pipeline origin are solid and point toward the origin if it can be expanded to show associated events. If the event information associated with an Pipeline origin can be collapsed, the arrow is open and points away from the origin. The `AddShippingAddress.pipeline` Pipeline origin in Figure 3-21 is an example of an origin that can be expanded; the `CalculateShippingCost.pipeline` Pipeline origin is an example of an origin that can be collapsed. Events are always represented by open diamonds.

Adding a Pipeline Origin

To add a Pipeline origin, follow these steps:

1. Click the plus (+) symbol located to the left of Add a Pipeline Origin. (If the list of Pipeline origins is long, you may need to scroll to the bottom of the page.) The Add Pipeline Origin page appears, as shown in Figure 3-22.

Figure 3-22 Add Pipeline Origin


Webflow Origin Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow Origin Editor
back
save
Document: Done

Add Pipeline

Enter the appropriate changes then click Save. Do not append the origin type to the origin name in the edit field.

Origin Name

Is Transactional

Component List

2. Enter the origin (Pipeline) name in the Origin Name text field.
3. If the Pipeline should be transactional, click the Is Transactional check box.

Notes: For more information about transactional and nontransactional Pipelines, see “Transactional Versus Nontransactional Pipelines” on page 4-11.

Initially, the Component List will be empty. You must add Pipeline components to the Pipeline separately, as described in “Adding a Pipeline Component” on page 3-42.

4. Click the Save button to save the new Pipeline origin. You are returned to the Pipeline Origins page, and the new origin appears alphabetically in the list.
5. Click the solid arrow located to the right of the new Pipeline origin to view its associated events. The arrow changes to an open arrow pointing in the opposite direction. The default event of `success()` → `index.jsp` (marked by an open diamond) and an option to Add an Event are shown.
6. If you want to add a different event to the Pipeline origin, modify the default event, or delete the default event, proceed to the appropriate section in “Modifying a Pipeline Origin’s Events” on page 3-53.

Note: Be sure to add components to the Pipeline origin, as described in “Adding a Pipeline Component” on page 3-42.

Editing a Pipeline Origin

You can edit a Pipeline origin in three ways:

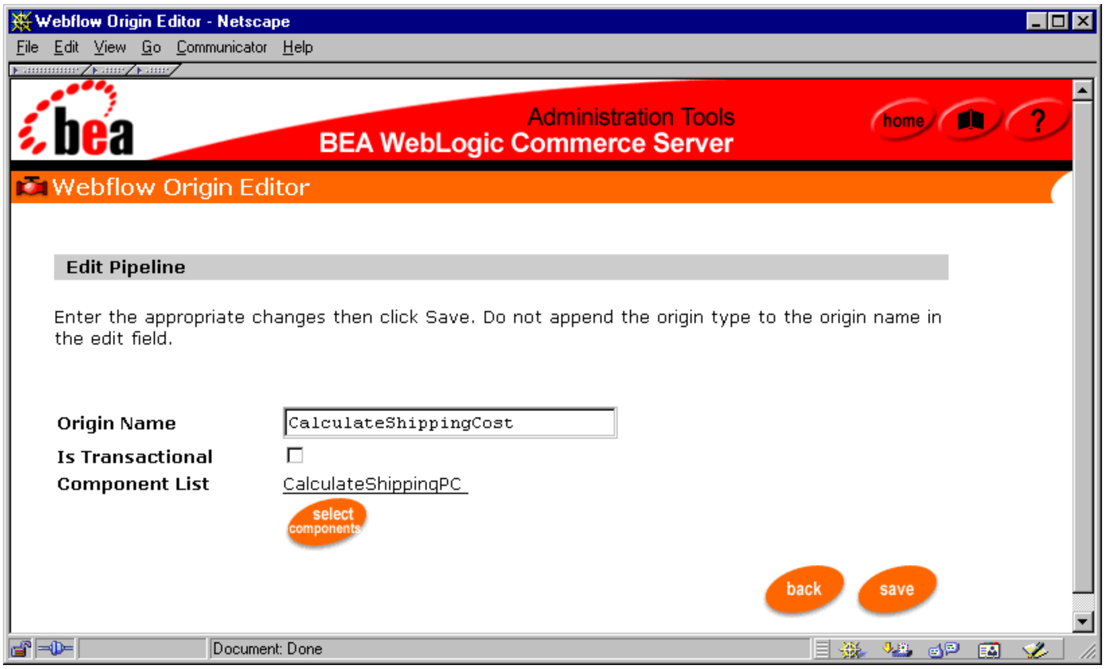
- By modifying information about the Pipeline origin itself (that is, its name or transactional status).
- By modifying the Pipeline origin’s components.
- By modifying the Pipeline origin’s associated events.

Modifying Information About a Pipeline Origin

To modify information about a Pipeline origin, follow these steps:

1. Click a hyperlinked Pipeline origin to edit it (for example, click `CalculateShippingCost.pipeline`). The Edit Pipeline Origin page appears, containing the original values for each field, as shown in Figure 3-23.

Figure 3-23 Edit Pipeline Origin


Webflow Origin Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow Origin Editor
select components
back
save
Document: Done

Edit Pipeline

Enter the appropriate changes then click Save. Do not append the origin type to the origin name in the edit field.

Origin Name	CalculateShippingCost
Is Transactional	☐
Component List	CalculateShippingPC

2. Change the origin (Pipeline) name or transactional state using the form fields.

3. Click the Save button to save your changes. You are returned to the Pipeline Origins page, and the modified Pipeline origin is shown alphabetically in the list.

4. If you want to modify the Pipeline origin's components, see "Modifying a Pipeline Origin's Components" on page 3-42.

Note: Modifications made to the Pipeline origin name and transaction status could potentially affect other areas of the Webflow that utilize it. Be sure your modifications take this into account, and be sure to verify your modified properties file with the validation tool. For more information about the validation tool, see "How to Validate Your Properties Files" on page 3-58.

Modifying a Pipeline Origin's Components

Recall that a Pipeline consists of a number of Pipeline components that perform specific tasks related to business processes. Pipeline components can be reordered, added, edited, or removed from the Pipeline.

Notes: For more information about Pipeline components, see "Customizing Pipelines Using the pipeline.properties File" on page 2-14.

Pipeline components can never be completely deleted from the Webflow and Pipeline Editor. They can only be removed from a Pipeline.

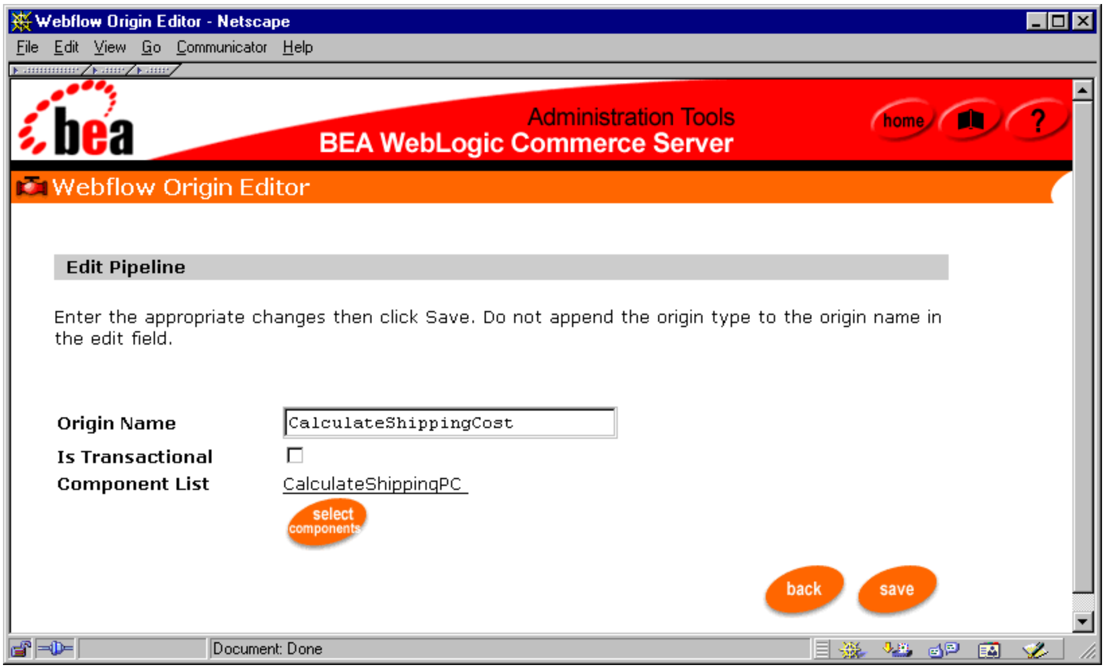
Adding a Pipeline Component

Once you have created a new Pipeline, you will want to add components to the Pipeline. To add Pipeline components to a Pipeline, follow these steps:

Note: If you just created a new Pipeline, you can begin at step 2.

1. Click a hyperlinked Pipeline origin to edit it (for example, click `CalculateShippingCost.pipeline`). The Edit Pipeline page appears, containing the original values for each field, as shown in Figure 3-24.

Figure 3-24 Edit Pipeline


Webflow Origin Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow Origin Editor

Edit Pipeline

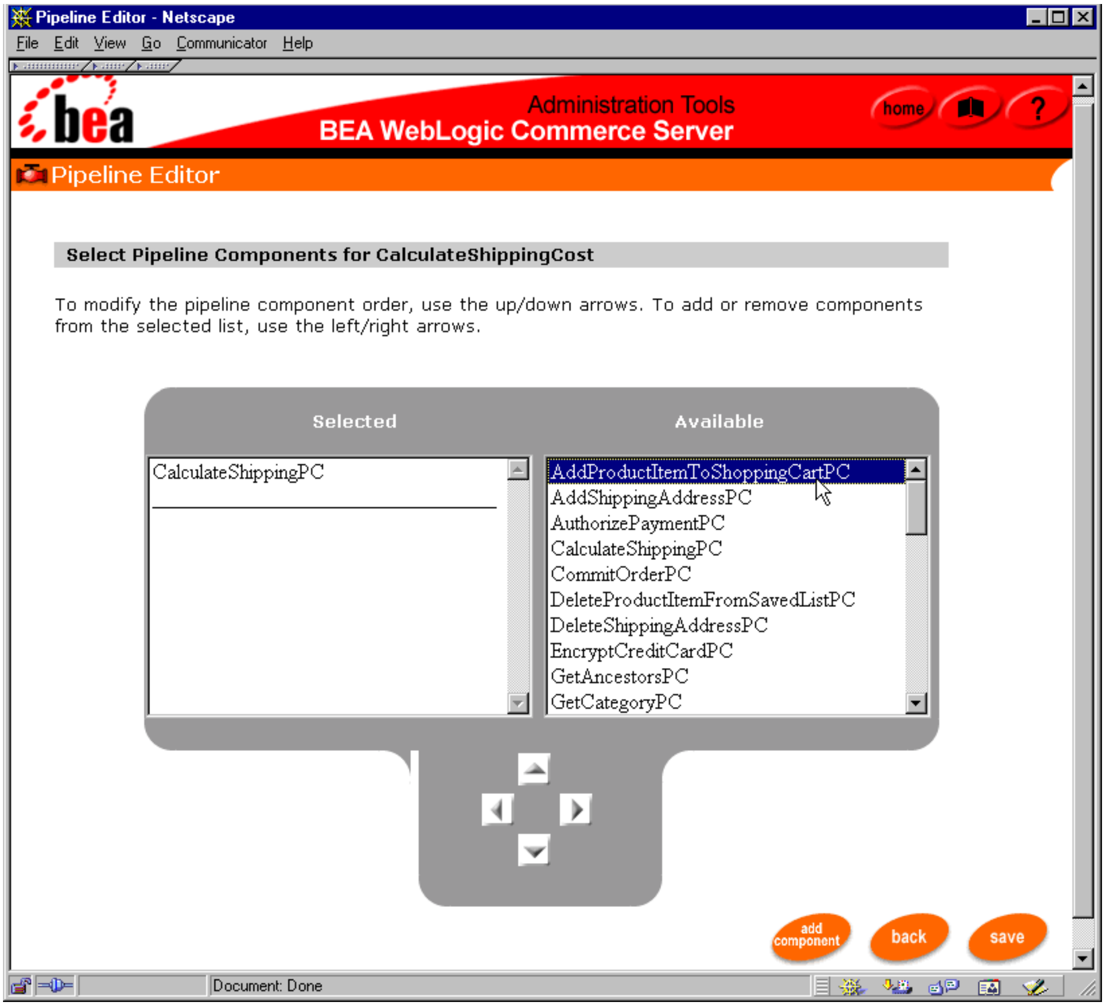
Enter the appropriate changes then click Save. Do not append the origin type to the origin name in the edit field.

Origin Name	CalculateShippingCost
Is Transactional	☐
Component List	CalculateShippingPC

select components
back
save
Document: Done

2. Click the Select Components button. The Select Pipeline Components page appears, as shown in Figure 3-25.

Figure 3-25 Select Pipeline Components


Pipeline Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Pipeline Editor

Select Pipeline Components for CalculateShippingCost

To modify the pipeline component order, use the up/down arrows. To add or remove components from the selected list, use the left/right arrows.

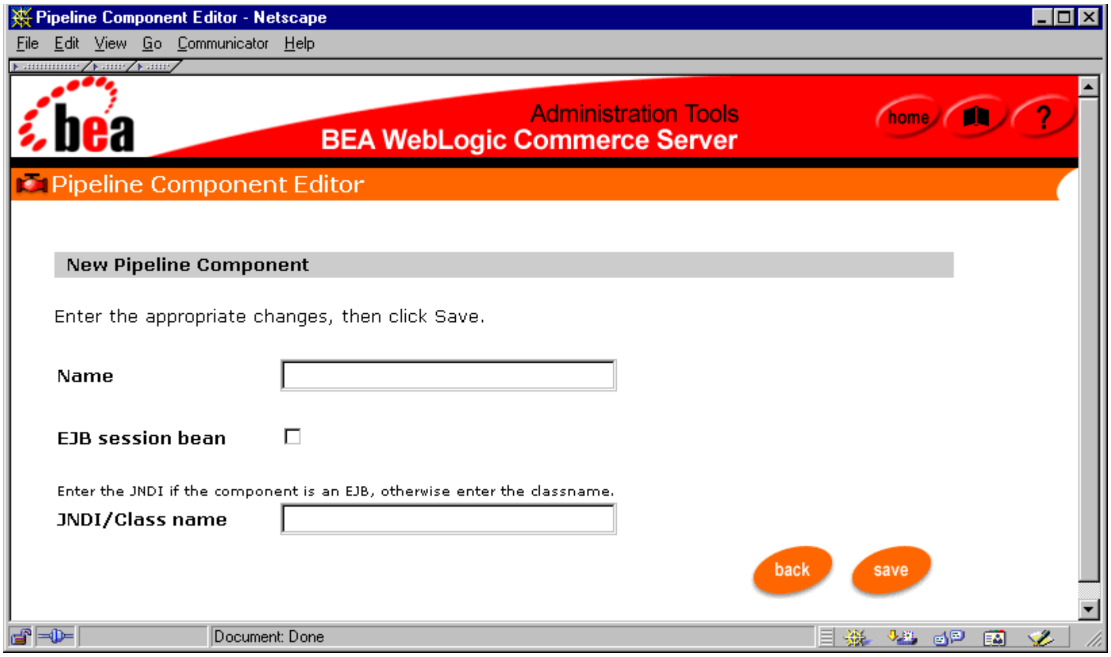
Selected	Available
CalculateShippingPC	AddProductItemToShoppingCartPC
	AddShippingAddressPC
	AuthorizePaymentPC
	CalculateShippingPC
	CommitOrderPC
	DeleteProductItemFromSavedListPC
	DeleteShippingAddressPC
	EncryptCreditCardPC
	GetAncestorsPC
	GetCategoryPC

3. If you want to add an existing Pipeline component to the Pipeline, select the component from the Available list and click the left arrow button to move it to the Selected list.

4. If you want to add a new Pipeline component to the Pipeline instead, follow these steps:

a. Click the Add Component button. The New Pipeline Component page appears, as shown in Figure 3-26.

Figure 3-26 New Pipeline Component


Pipeline Component Editor - Netscape
File Edit View Go Communicator Help
Administration Tools
BEA WebLogic Commerce Server
home
?
Pipeline Component Editor
back
save
Document: Done

New Pipeline Component

Enter the appropriate changes, then click Save.

Name

EJB session bean ☐

Enter the JNDI if the component is an EJB, otherwise enter the classname.

JNDI/Class name

b. Enter the Pipeline component name in the Name text field.

c. If the Pipeline component is implemented as an EJB session bean, click the EJB Session Bean check box.

d. If you checked the EJB Session Bean check box in the previous step, provide the JNDI name in the JNDI/Class Name text field. If you did not check the EJB Session Bean check box, enter the class name in the field instead.

e. Click the Save button to save your changes. You are returned to the Select Pipeline Components page, and the new Pipeline component is shown *at the bottom* of the Available List.

f. Select the new Pipeline component from the Available list and click the left arrow button to move it to the Selected list.

5. Click the Save button to save your changes. You are returned to the Edit Pipeline page, and the Pipeline component you added appears as a hyperlink to the left of the Component List label.

Notes: Pipeline components appear on the Edit Pipeline page in the order they will be executed in the Pipeline. For information about reordering components in the Pipeline, see “Reordering Pipeline Components” on page 3-46.

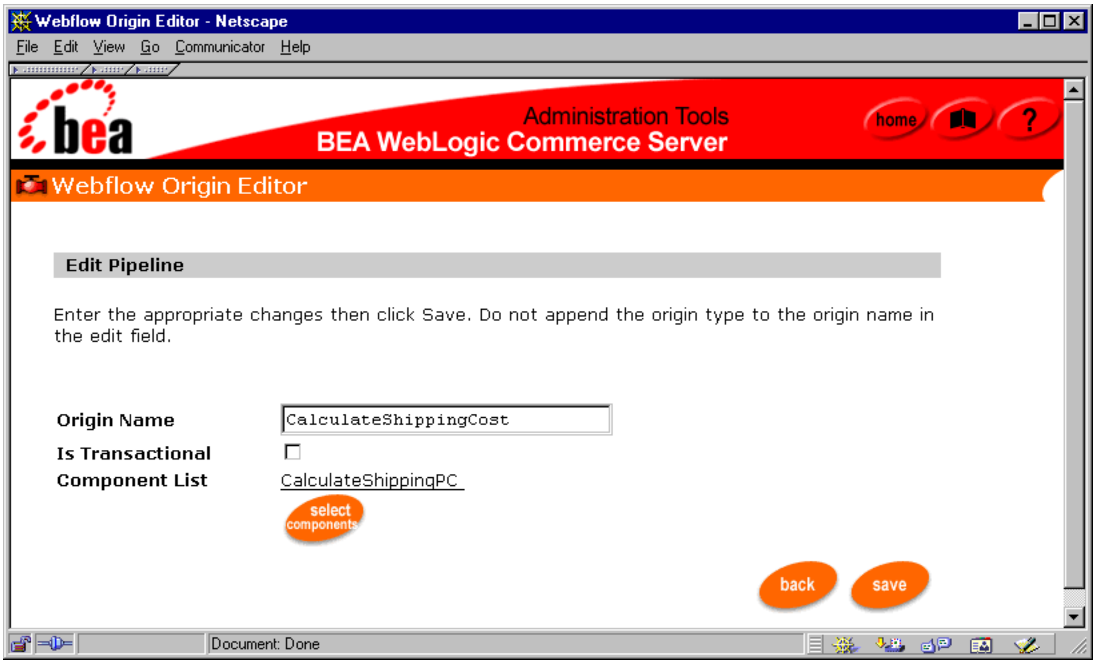
If you make other changes to the Pipeline on the Edit Pipeline page, you will need to click the Save button on this page as well.

Reordering Pipeline Components

Occasionally, you may want to reorder the components in a Pipeline. To reorder Pipeline components within a Pipeline, follow these steps:

1. Click a hyperlinked Pipeline origin to edit it (for example, click `CalculateShippingCost.pipeline`). The Edit Pipeline page appears, containing the original values for each field, as shown in Figure 3-27.

Figure 3-27 Edit Pipeline


Webflow Origin Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow Origin Editor

Edit Pipeline

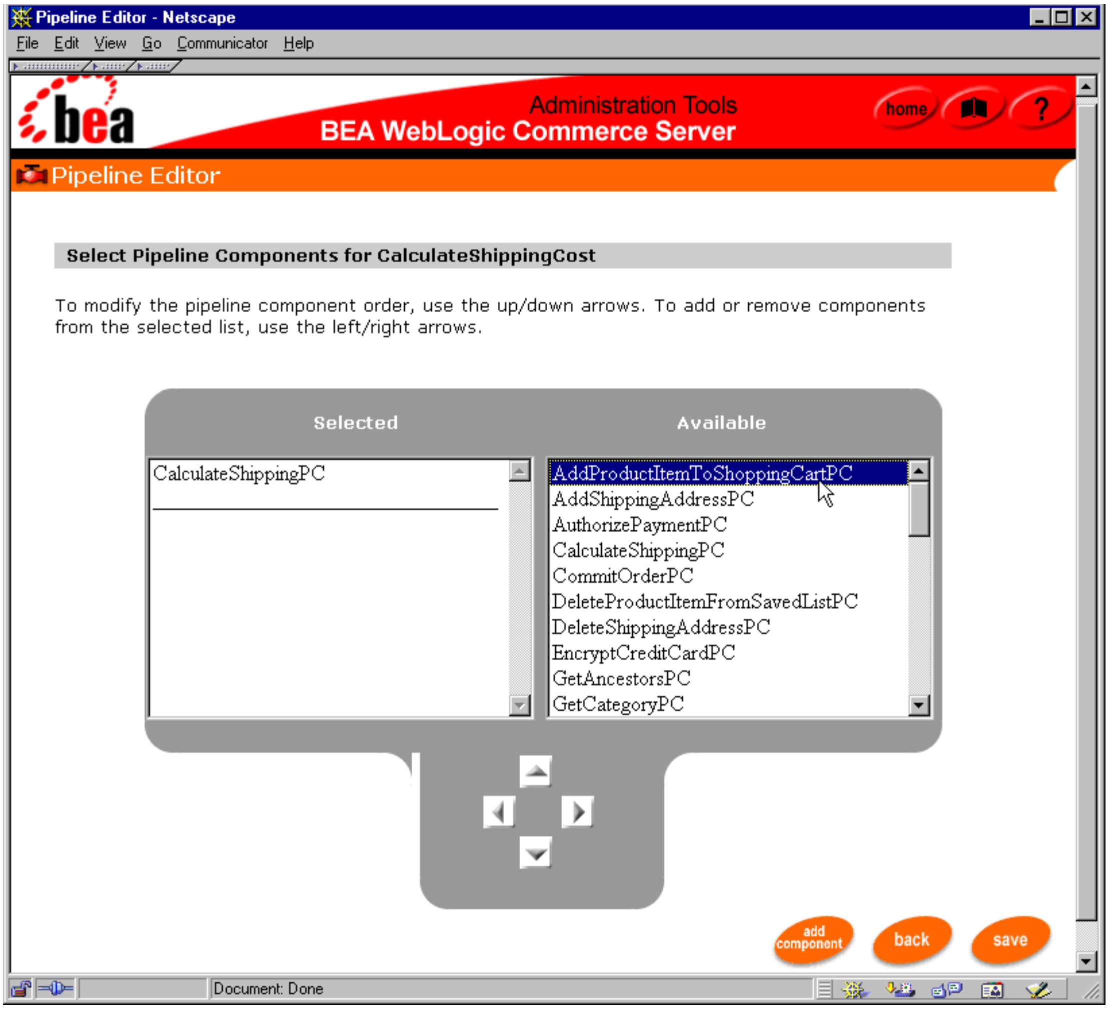
Enter the appropriate changes then click Save. Do not append the origin type to the origin name in the edit field.

Origin Name	CalculateShippingCost
Is Transactional	☐
Component List	CalculateShippingPC

select components
back
save
Document: Done

2. Click the Select Components button. The Select Pipeline Components page appears, as shown in Figure 3-28.

Figure 3-28 Select Pipeline Components



Select Pipeline Components for CalculateShippingCost

To modify the pipeline component order, use the up/down arrows. To add or remove components from the selected list, use the left/right arrows.

Selected: CalculateShippingPC

Available: AddProductItemToShoppingCartPC, AddShippingAddressPC, AuthorizePaymentPC, CalculateShippingPC, CommitOrderPC, DeleteProductItemFromSavedListPC, DeleteShippingAddressPC, EncryptCreditCardPC, GetAncestorsPC, GetCategoryPC

3. Click the Pipeline component in the Selected list to highlight it.

4. Use the up or down arrow buttons to move the Pipeline component up or down in the Selected list.

Note: You can add a Pipeline component to more than one Pipeline, so the Pipeline component you moved to the Selected list will still appear in the Available list.

5. Click the Save button to save your changes. You are returned to the Edit Pipeline page, and the Pipeline component appears as a hyperlink to the left of the Component List label, in the order you specified.

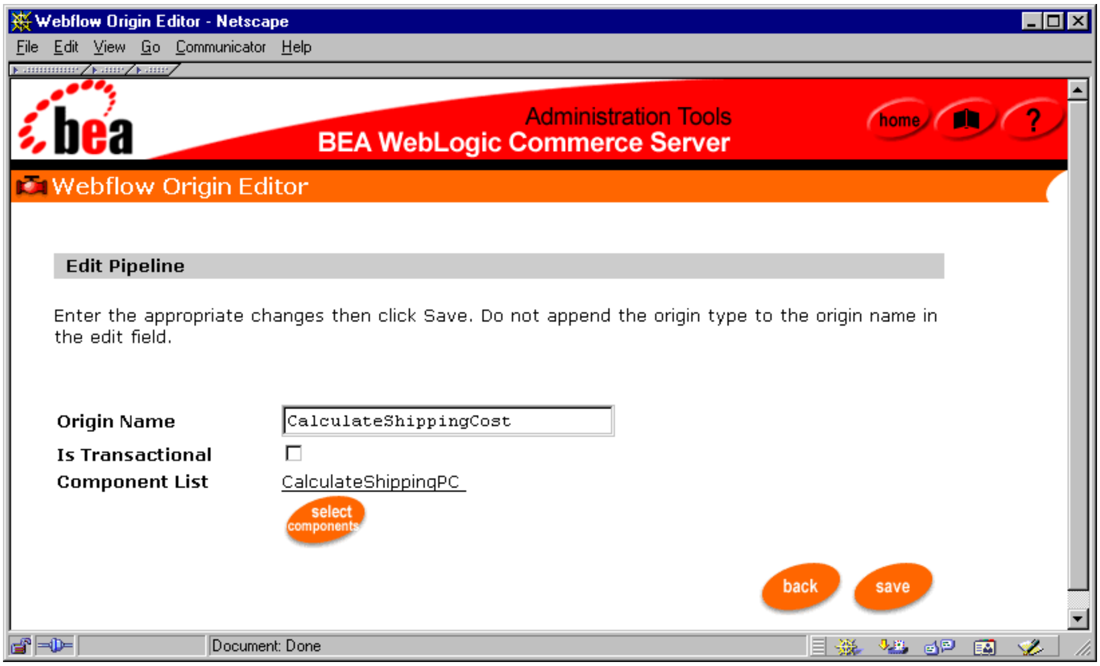
Note: If you make other changes to the Pipeline on the Edit Pipeline page, you will need to click the Save button on this page as well.

Editing a Pipeline Component

Occasionally, you may want to edit a component that already exists within a Pipeline. To edit a Pipeline component, follow these steps:

1. Click a hyperlinked Pipeline origin to edit it (for example, click `CalculateShippingCost.pipeline`). The Edit Pipeline page appears, containing the original values for each field, as shown in Figure 3-29.

Figure 3-29 Edit Pipeline


Webflow Origin Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow Origin Editor

Edit Pipeline

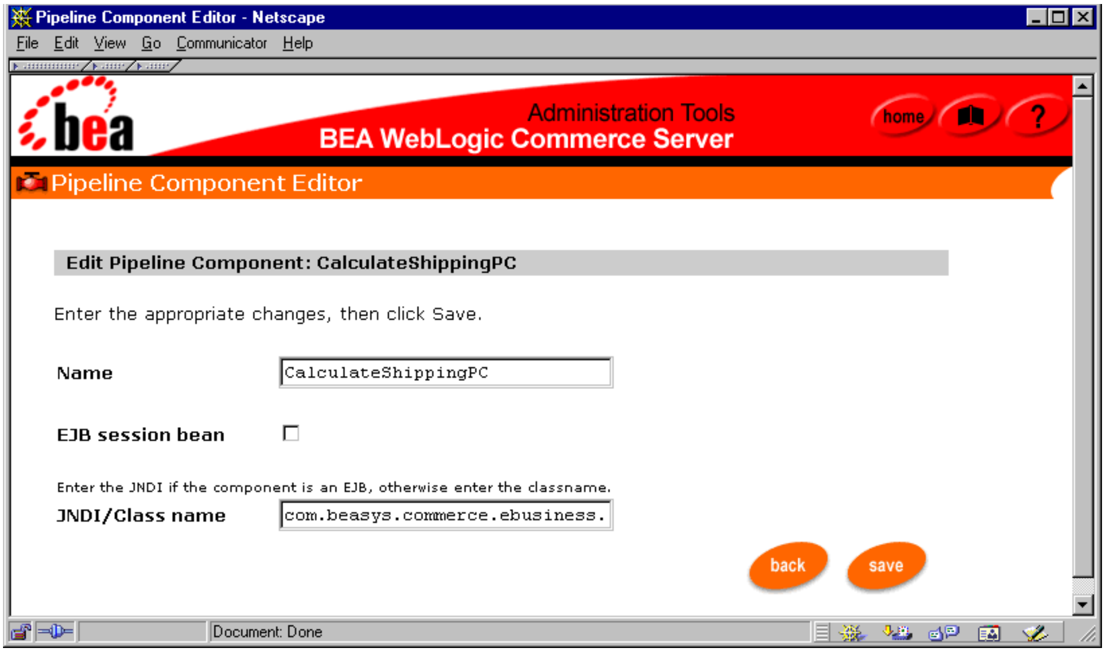
Enter the appropriate changes then click Save. Do not append the origin type to the origin name in the edit field.

Origin Name	CalculateShippingCost
Is Transactional	☐
Component List	CalculateShippingPC

select components
back
save
Document: Done

2. Click the hyperlinked Pipeline component name, which appears in a list to the left of the Component List label. The Edit Pipeline Component page appears, as shown in Figure 3-30.

Figure 3-30 Edit Pipeline Component


Pipeline Component Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Pipeline Component Editor
back
save
Document: Done

Edit Pipeline Component: CalculateShippingPC

Enter the appropriate changes, then click Save.

Name: CalculateShippingPC

EJB session bean: ☐

Enter the JNDI if the component is an EJB, otherwise enter the classname.

JNDI/Class name: com.beasys.commerce.ebusiness.

3. Use the Name text field, EJB Session Bean check box, and JNDI/Class Name text field to make your modifications.

4. Click the Save button to save your changes. You are returned to the Edit Pipeline page, and the Pipeline component you edited still appears as a hyperlink to the left of the Component List label.

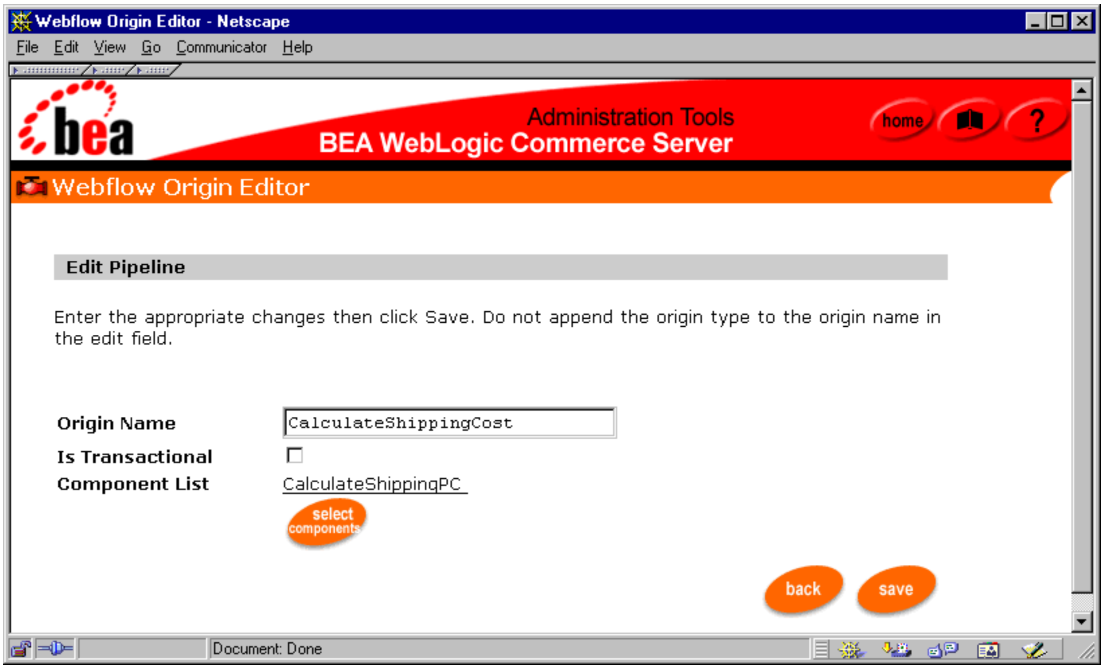
Note: If you make other changes to the Pipeline on the Edit Pipeline page, you will need to click the Save button on this page as well.

Removing a Pipeline Component

Occasionally, you may want to remove a component from a Pipeline. To remove a Pipeline component from a Pipeline, follow these steps:

1. Click a hyperlinked Pipeline origin to edit it (for example, click `CalculateShippingCost.pipeline`). The Edit Pipeline page appears, containing the original values for each field, as shown in Figure 3-31.

Figure 3-31 Edit Pipeline


Webflow Origin Editor - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow Origin Editor

Edit Pipeline

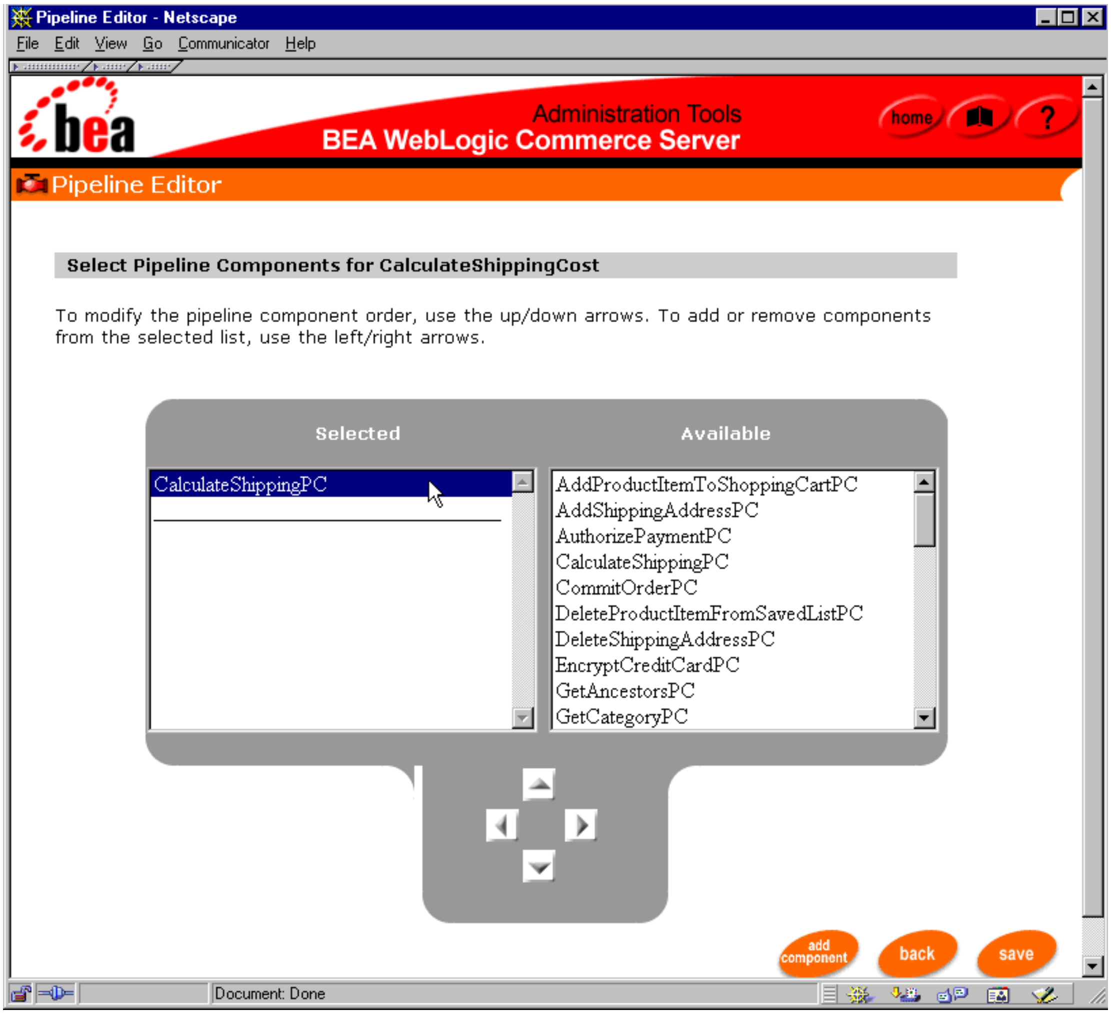
Enter the appropriate changes then click Save. Do not append the origin type to the origin name in the edit field.

Origin Name	CalculateShippingCost
Is Transactional	☐
Component List	CalculateShippingPC

select components
back
save
Document: Done

2. Click the Select Components button. The Select Pipeline Components page appears, as shown in Figure 3-32.

Figure 3-32 Select Pipeline Components



Select Pipeline Components for CalculateShippingCost

To modify the pipeline component order, use the up/down arrows. To add or remove components from the selected list, use the left/right arrows.

Selected	Available
CalculateShippingPC	AddProductItemToShoppingCartPC
	AddShippingAddressPC
	AuthorizePaymentPC
	CalculateShippingPC
	CommitOrderPC
	DeleteProductItemFromSavedListPC
	DeleteShippingAddressPC
	EncryptCreditCardPC
	GetAncestorsPC
	GetCategoryPC

3. Click the Pipeline component in the Selected list to highlight it.

4. Use the right arrow button to move the Pipeline component from the Selected list to the Available list.

5. Click the Save button to save your changes. You are returned to the Edit Pipeline page, and the Pipeline component you removed no longer appears as a hyperlink to the left of the Component List label.

Note: If you make other changes to the Pipeline on the Edit Pipeline page, you will need to click the Save button on this page as well.

Modifying a Pipeline Origin's Events

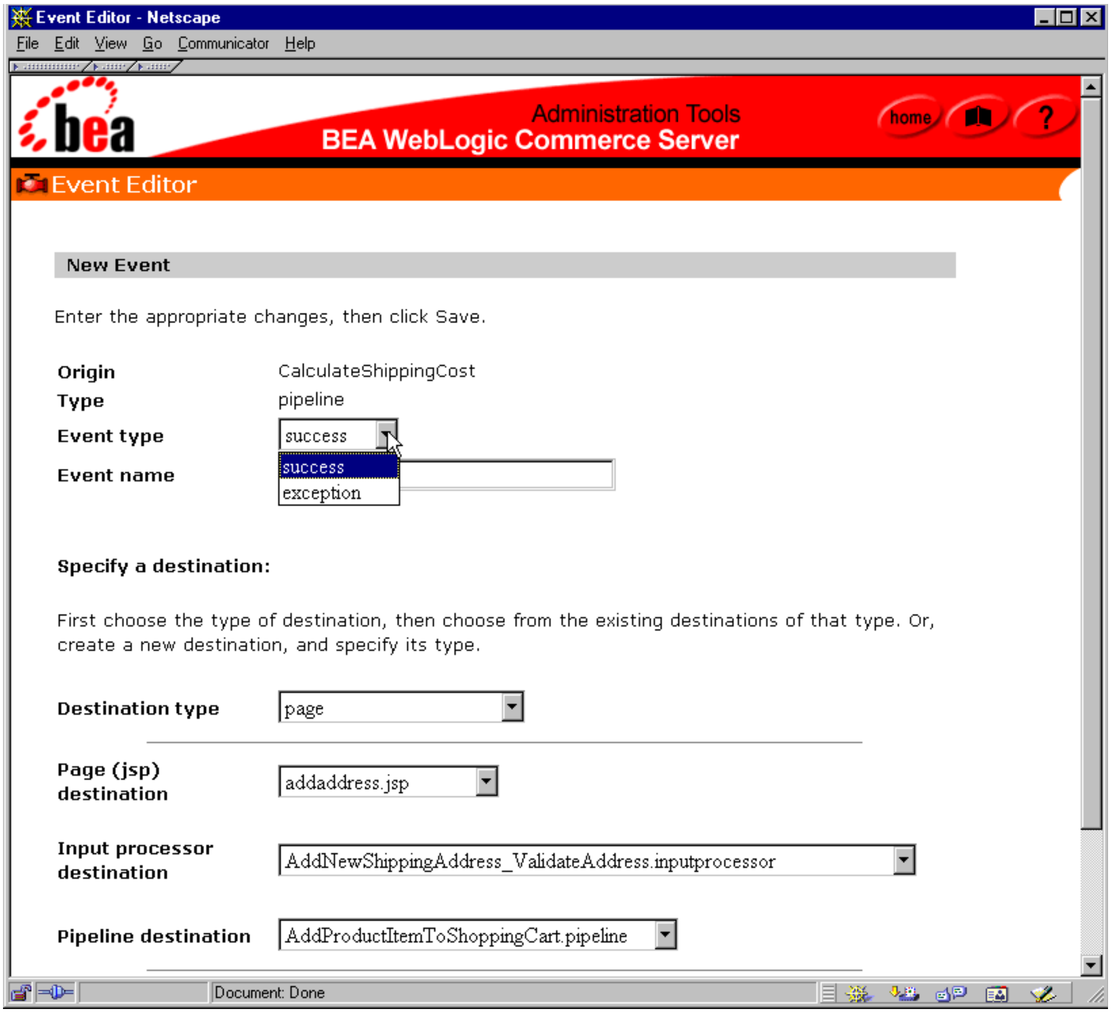
You can also edit a Pipeline origin by modifying its associated events. You can add, edit, or delete events associated with a Pipeline origin as described in the following sections.

Adding an Event

To add an event to a Pipeline origin, follow these steps:

1. Verify that the Pipeline origin to which you want to add the event is expanded (that is, an open arrow is shown to the left of the Pipeline origin, and the Add an Event option is shown). If the Pipeline origin is not expanded, click the solid arrow located to the left of the Pipeline origin to expand it.
2. Click the plus (+) sign located next to Add an Event. (If the list of events is long, you may need to use the scrollbar.) The New Event page appears, as shown in Figure 3-33.

Figure 3-33 Pipeline Origin: New Event



New Event

Enter the appropriate changes, then click Save.

Origin	CalculateShippingCost
Type	pipeline
Event type	success (success, exception)
Event name	

Specify a destination:

First choose the type of destination, then choose from the existing destinations of that type. Or, create a new destination, and specify its type.

Destination type	page
Page (jsp) destination	addaddress.jsp
Input processor destination	AddNewShippingAddress_ValidateAddress.inputprocessor
Pipeline destination	AddProductItemToShoppingCart.pipeline

3. Select the appropriate Event Type from the pull-down menu, and enter a name for the event in the Event Name text field.

Note: For more information about event types and event names, see “Syntax of the webflow.properties File” on page 2-3 and “About Event Names” on page 2-4, respectively.

4. Next, you need to specify a destination for the event. The destination could be an existing page (`.htm`, `.html`, or `.jsp`), input processor, or Pipeline, or the destination could be an entirely new page, input processor, or Pipeline. To specify the destination, follow these steps:

   a. Select the destination type from the Destination Type pull-down menu (one of Page, Input Processor, Pipeline, or Specify New Destination).

Note: Selecting a destination type in this menu disables form fields that are not required for the chosen destination type. To enable unavailable form fields, choose a different destination type.

   b. If you selected Page from the Destination Type pull-down menu, select an existing page (`.htm`, `.html`, or `.jsp`) from the Page (JSP) Destination pull-down menu. If you selected Input Processor, select an existing input processor from the Input Processor Destination pull-down menu. If you selected Pipeline, select an existing Pipeline from the Pipeline Destination pull-down menu. If you selected Specify New Destination from the Destination Type pull-down menu, enter the name of the destination in the Specify New Destination text field, and select the destination type from the Of Type pull-down menu (jsp, htm, html, inputprocessor, or pipeline).

   c. Click the Save button to save the new event. You are returned to the Pipeline Origins page, and the new event (marked by an open diamond) appears beneath the appropriate Pipeline origin in an alphabetic list.

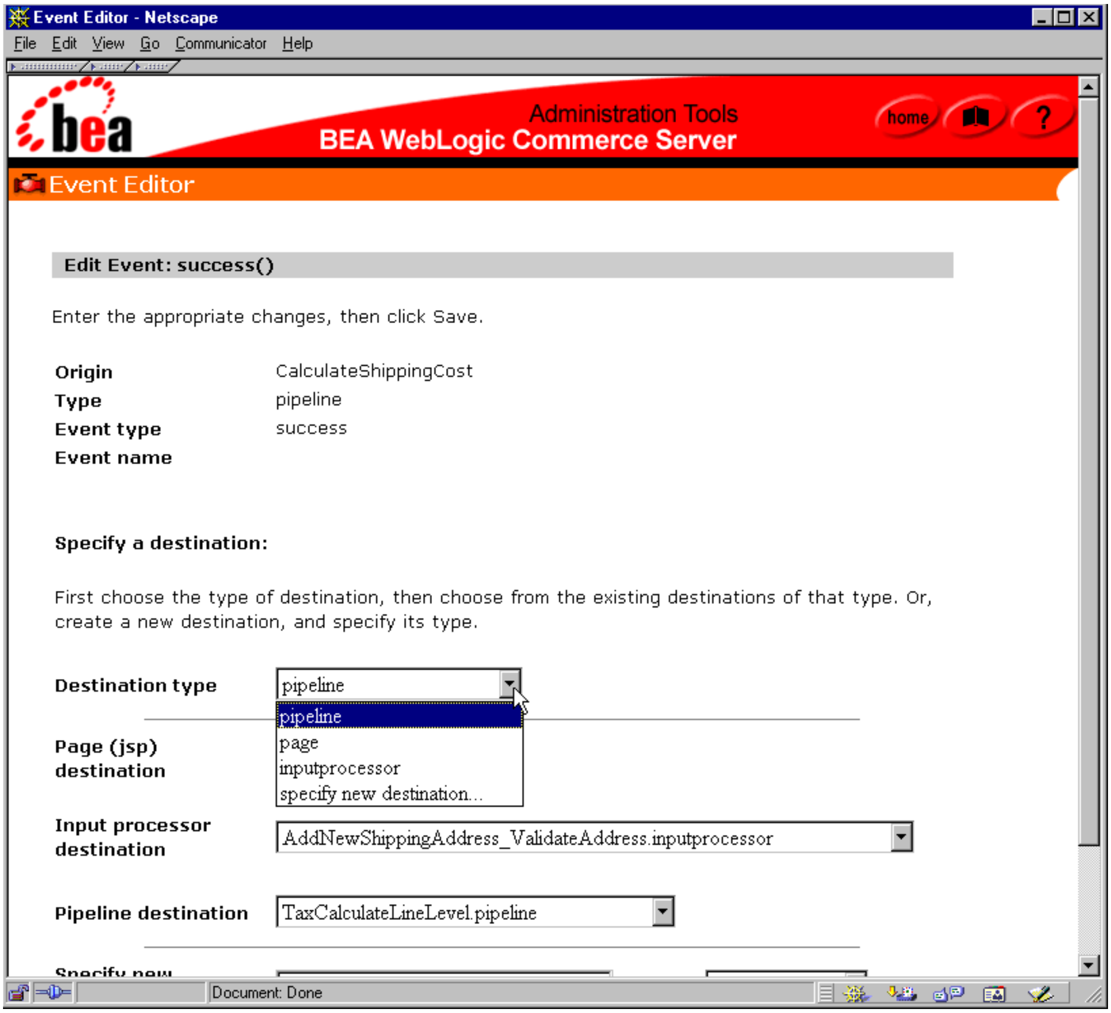
Editing an Event

To edit an event already associated with a Pipeline origin, follow these steps:

1. Verify that the Pipeline origin for which you want to edit an event is expanded (that is, an open arrow is shown to the left of the Pipeline origin, and the event you want to modify is shown). If the Pipeline origin is not expanded, click the solid arrow located to the left of the Pipeline origin to expand it.

2. Click a hyperlinked event to edit it (for example, click `success()` → `TaxCalculateLineLevel.pipeline`). The Edit Event page appears, containing the original values for each field, as shown in Figure 3-34.

Figure 3-34 Pipeline Origin: Edit Event



Edit Event: success()

Enter the appropriate changes, then click Save.

Origin	CalculateShippingCost
Type	pipeline
Event type	success
Event name	

Specify a destination:

First choose the type of destination, then choose from the existing destinations of that type. Or, create a new destination, and specify its type.

Destination type	pipeline (pipeline, page, inputprocessor, specify new destination...)
Page (jsp) destination	
Input processor destination	AddNewShippingAddress_ValidateAddress.inputprocessor
Pipeline destination	TaxCalculateLineLevel.pipeline

Specify new

Note: The Edit Event page does not allow you to modify the event type or event name. To do so, you must first delete the event using the instructions provided in “Deleting an Event” on page 3-57 and then recreate it using the instructions provided in “Adding an Event” on page 3-53.

3. Recall that an event destination could be an existing page (`.htm`, `.html`, `.jsp`), input processor, or Pipeline, or an entirely new page, input processor, or Pipeline. To change the destination for the event, follow these steps:

   a. Select the destination type from the Destination Type pull-down menu (one of Page, Input Processor, Pipeline, or Specify New Destination).

Note: Selecting a destination type in this menu disables form fields that are not required for the chosen destination type. To enable unavailable form fields, choose a different destination type.

   b. If you selected Page from the Destination Type pull-down menu, select an existing page (`.htm`, `.html`, or `.jsp`) from the Page (JSP) Destination pull-down menu. If you selected Input Processor, select an existing input processor from the Input Processor Destination pull-down menu. If you selected Pipeline, select an existing Pipeline from the Pipeline Destination pull-down menu. If you selected Specify New Destination from the Destination Type pull-down menu, enter the name of the destination in the Specify New Destination text field, and select the destination type from the Of Type pull-down menu (jsp, htm, html, inputprocessor, or pipeline).

   c. Click the Save button to save the changes to the event. You are returned to the Pipeline Origins page, and the modified event (marked by an open diamond) appears beneath the appropriate Pipeline origin in an alphabetic list.

Deleting an Event

To delete an event currently associated with a Pipeline origin, follow these steps:

1. Verify that the Pipeline origin for which you want to delete an event is expanded (that is, an open arrow is shown to the left of the Pipeline origin, and the event you want to delete is shown). If the Pipeline origin is not expanded, click the solid arrow located to the left of the Pipeline origin to expand it.

2. To delete the event, click the red X located to the right of the event.

3. Click the OK button in the confirmation pop-up window.

Note: If you delete all the events associated with a Pipeline origin and then leave the Pipeline Origins page (or click the arrow next to the origin to collapse it), the origin itself will also be deleted. This is done to prevent an error condition in the Webflow. However, if you just delete all the events associated with a

Pipeline origin and immediately start to add new events to the Pipeline origin, the origin itself will remain. This way, you do not always have to recreate the origin after deleting all of its events.

Deleting a Pipeline Origin

To delete a Pipeline origin, click the red X located to the right of the origin. Confirm the deletion by clicking the OK button.

Warning: Deleting a Pipeline origin means that the origin and all of its associated events will be deleted.

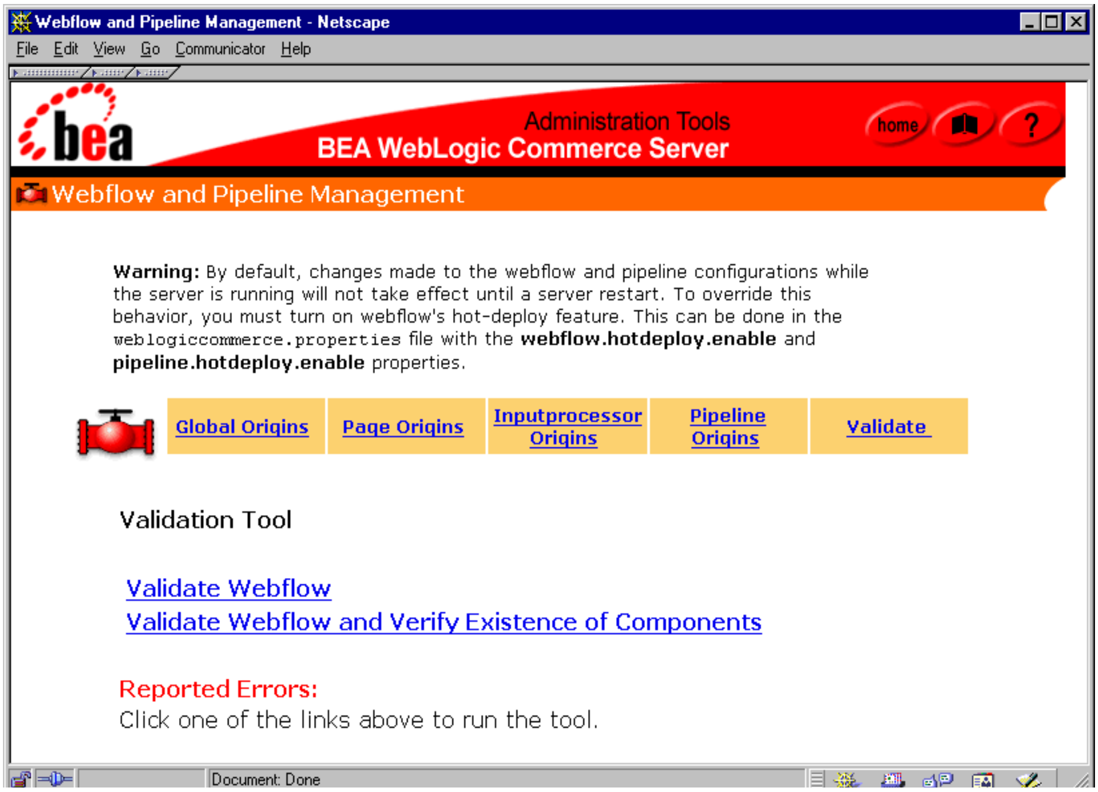
How to Validate Your Properties Files

The Webflow and Pipeline Editor provides you with two methods for validating your Webflow and Pipeline properties files. Using the validation tool in the Webflow and Pipeline Editor, you can either:

- Validate the Webflow, or
- Validate the Webflow and verify the existence of various Webflow components (that is, input processors, Pipeline components, and so on).

To open the Webflow and Pipeline Editor validation tool, click the Validate link (or its corresponding Validate button). The Validation Tool page appears, as shown in Figure 3-35.

Figure 3-35 Validation Tool


Webflow and Pipeline Management - Netscape
File Edit View Go Communicator Help
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow and Pipeline Management

Warning: By default, changes made to the webflow and pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on webflow's hot-deploy feature. This can be done in the `weblogiccommerce.properties` file with the **webflow.hotdeploy.enable** and **pipeline.hotdeploy.enable** properties.

Global Origins
Page Origins
Inputprocessor Origins
Pipeline Origins
Validate

Validation Tool

Validate Webflow
Validate Webflow and Verify Existence of Components

Reported Errors:
Click one of the links above to run the tool.

The remainder of this section describes how to use the two options in the Webflow and Pipeline Editor validation tool, and provides some helpful information about interpreting reported errors.

Validating the Webflow

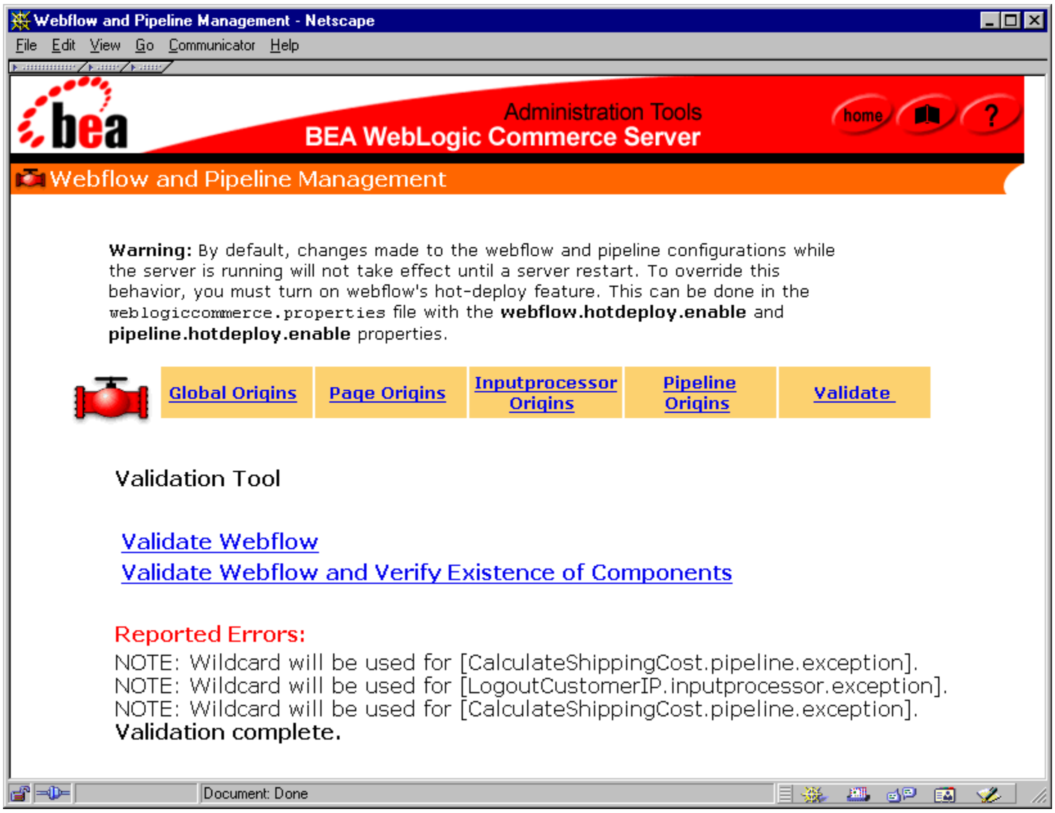
Validation of Webflow syntax includes checks for the following:

- Missing states (that is, `success`, `exception`).
- Wildcards (in the case of missing states).
- Proper configuration of the `configurationerrorpage` property.

- Validity of extensions (valid extensions are `.jsp`, `.htm`, `.html`, `.inputprocessor` and `.pipeline`).
- Missing configurations (for example, references to nonexistent `home.jsp.link(browse)`).
- Missing class names for `Inputprocessors`.
- Validity of Pipeline definitions (that is, a Pipeline's `isTransactional` property and component list).

To validate only the Webflow syntax, click the Validate Webflow link. Any problems with the Webflow will appear at the bottom of the page under the Reported Errors label, as shown in Figure 3-36.

Figure 3-36 Webflow Validation


Webflow and Pipeline Management - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow and Pipeline Management
Global Origins
Page Origins
Inputprocessor Origins
Pipeline Origins
Validate
Document: Done

Warning: By default, changes made to the webflow and pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on webflow's hot-deploy feature. This can be done in the `weblogiccommerce.properties` file with the **webflow.hotdeploy.enable** and **pipeline.hotdeploy.enable** properties.

Validation Tool

Validate Webflow
Validate Webflow and Verify Existence of Components

Reported Errors:

```
NOTE: Wildcard will be used for [CalculateShippingCost.pipeline.exception].
NOTE: Wildcard will be used for [LogoutCustomerIP.inputprocessor.exception].
NOTE: Wildcard will be used for [CalculateShippingCost.pipeline.exception].
Validation complete.
```

Note: In addition to errors, the output produced by the validation tool may contain notes or warnings. These are not error conditions per se, but are informational in nature. For more information about the validation tool's output, see "Validator Message Descriptions" on page 3-63.

If any errors are shown, use the other options in the Webflow and Pipeline Editor to make modifications. It is a good idea to run the validation tool until there are no errors in your Webflow.

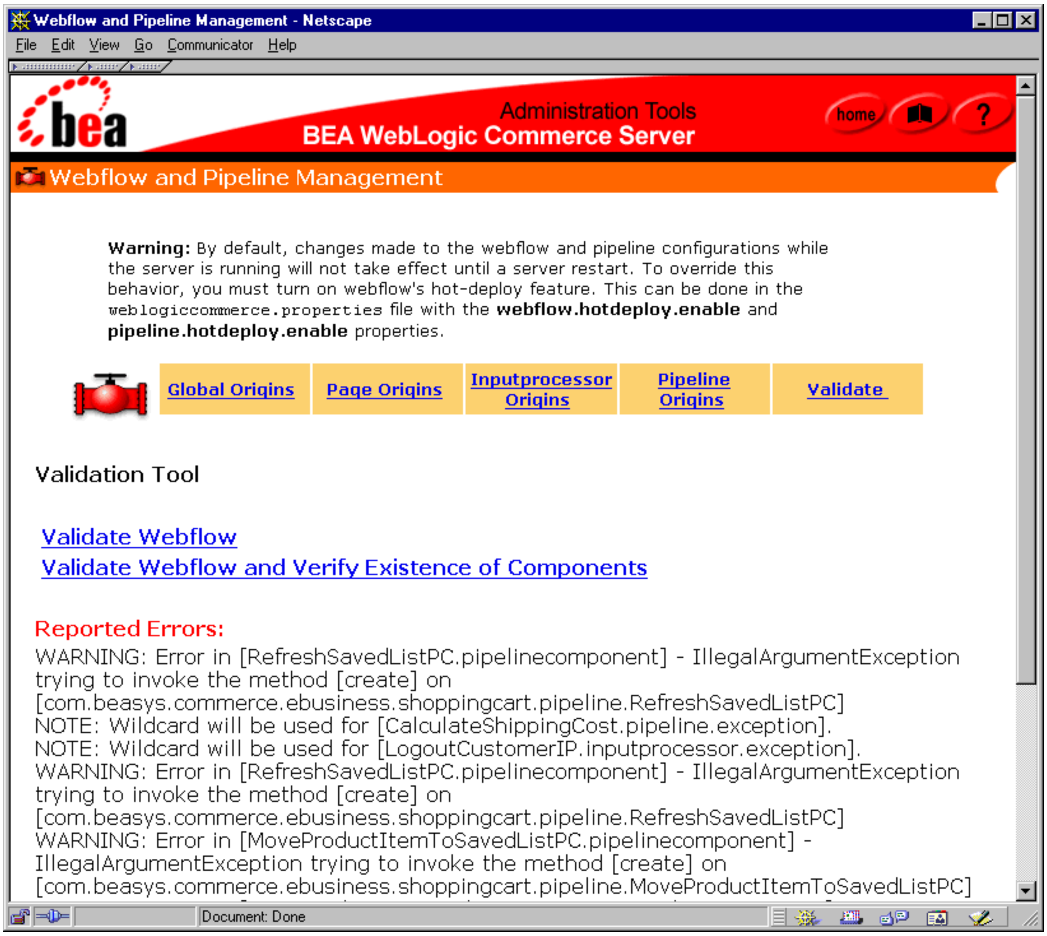
Validating the Webflow and Verifying the Existence of Components

Validation of Webflow syntax with verification of components includes checks for the following:

- Missing states (that is, `success`, `exception`).
- Wildcards (in the case of missing states).
- Proper configuration of the `configurationerrorpage` property.
- Validity of extensions (valid extensions are `.jsp`, `.htm`, `.html`, `.inputprocessor` and `.pipeline`).
- Missing configurations (for example, references to non-existent `home.jsp.link(browse)`).
- Missing class names for `Inputprocessors`.
- Validity of Pipeline definitions (that is, a Pipeline's `isTransactional` property and component list).
- Deployment (existence) of `InputProcessors` and `PipelineComponents`.

To validate the Webflow and verify the existence of components, click the Validate Webflow and Verify Existence of Components link. Any problems with the Webflow and its components will appear at the bottom of the page under the Reported Errors label, as shown in Figure 3-37.

Figure 3-37 Webflow Validation and Verification of Components


Webflow and Pipeline Management - Netscape
File Edit View Go Communicator Help
bea
Administration Tools
BEA WebLogic Commerce Server
home
?
Webflow and Pipeline Management

Warning: By default, changes made to the webflow and pipeline configurations while the server is running will not take effect until a server restart. To override this behavior, you must turn on webflow's hot-deploy feature. This can be done in the `weblogiccommerce.properties` file with the **webflow.hotdeploy.enable** and **pipeline.hotdeploy.enable** properties.

Global Origins
Page Origins
Inputprocessor Origins
Pipeline Origins
Validate

Validation Tool

Validate Webflow
Validate Webflow and Verify Existence of Components

Reported Errors:

```
WARNING: Error in [RefreshSavedListPC.pipelinecomponent] - IllegalArgumentException
trying to invoke the method [create] on
[com.beasys.commerce.ebusiness.shoppingcart.pipeline.RefreshSavedListPC]
NOTE: Wildcard will be used for [CalculateShippingCost.pipeline.exception].
NOTE: Wildcard will be used for [LogoutCustomerIP.inputprocessor.exception].
WARNING: Error in [RefreshSavedListPC.pipelinecomponent] - IllegalArgumentException
trying to invoke the method [create] on
[com.beasys.commerce.ebusiness.shoppingcart.pipeline.RefreshSavedListPC]
WARNING: Error in [MoveProductItemToSavedListPC.pipelinecomponent] -
IllegalArgumentException trying to invoke the method [create] on
[com.beasys.commerce.ebusiness.shoppingcart.pipeline.MoveProductItemToSavedListPC]
```

Note: In addition to errors, the output produced by the validation tool may contain notes or warnings. These are not error conditions per se, but are informational in nature. For more information about the validation tool's output, see "Validator Message Descriptions" on page 3-63.

If any errors are shown, use the other options in the Webflow and Pipeline Editor to make modifications. It is a good idea to run the validation tool until there are no errors in your Webflow.

Validator Message Descriptions

The message descriptions should assist you in resolving issues identified by the Webflow and Pipeline Editor validation tool.

Notes

A message with a NOTE is displayed if a wildcard is used in place of a missing configuration.

Warnings

A WARNING is displayed when there are missing states or missing exceptions for an `InputProcessor` or a `Pipeline`.

Errors

An ERROR is displayed when there are: missing or invalid extensions; missing configurations, Pipeline definitions, `classNames` (for `InputProcessors`) or `jndiNames` (for `PipelineComponents`). Additionally, an ERROR can be displayed if the validation tool is unable to verify the existence of either an `InputProcessor` or a `PipelineComponent`.

CHAPTER

4 Extending Webflow and Pipelines

Although the BEA WebLogic Commerce Server product provides default Webflow and Pipeline mechanisms that you can customize, the Webflow and Pipelines have also been designed for easy extensibility. For example, if your organizational requirements dictate the use of a new business process, the Java/EJB programmers on your development team can utilize the existing Webflow and Pipeline infrastructure to create and incorporate these components into the system. This topic describes how to accomplish this.

This topic includes the following sections:

- Using the PipelineComponent Interface
- Pipeline Component Exceptions
- The CommercePipelineComponent Base Class
- Pipeline Component Naming Conventions
- Implementation of Pipeline Components as Stateless Session EJBs or Java Objects
- Stateful Versus Stateless Pipeline Components
- Transactional Versus Nontransactional Pipelines
- Other Development Guidelines

■ Handling Session Timeouts

- Using the getPipelineSession() Method
- The InvalidSessionStateException Exception in webflow.properties
- PipelineComponent and Session Timeouts
- The InvalidPipelineSessionStateException Exception in webflow.properites
- About the sessiontimeout.jsp Template

Pipeline Sessions

Although Pipelines and their components are reusable, they must relate to a particular customer's experience on your e-commerce site to make their execution relevant. For this reason, Pipeline components always operate on a Pipeline session. This section provides you with information about the Pipeline session, and provides instructions for configuring the Pipeline session to meet your own needs.

What Is a Pipeline Session?

Clearly, it is necessary to keep track of information gathered from your customers and the data modified by Pipeline components as a customer moves through your site. To maintain this state of the business process, the BEA WebLogic Commerce Server product makes use of a Pipeline session. A Pipeline session is an object that is created and stored within the HTTP session, with the goal of providing a single point of communication for all Pipeline components in a given Pipeline. Additionally, Pipeline sessions provide central access and storage for all external classes that may also need to update the Pipeline session.

The Pipeline session is comprised of many name/value pairs called *attributes*. Pipeline components act on particular attributes that exist within the Pipeline session, and may also add new attributes as necessary.

Attribute Scoping

The Pipeline session provides an API that allows you to add Pipeline session attributes. All attributes in the Pipeline session can have one of two scopes: Pipeline Session scope or Request scope. The method signature for creating Pipeline session attributes is:

```
public void setAttribute(String key, Object attribute, int scope);
```

where `scope` is either `PipelineConstants.PIPELINE_SESSION_SCOPE` or `PipelineConstants.REQUEST_SCOPE`.

In the Pipeline Session scope, the attribute exists in the Pipeline session until the end of the current HTTP session. Pipeline Session scope is the default scope for Pipeline session attributes, and will be used if the third parameter to the `setAttribute()` method is not specified. In the Request scope, the attributes are made available in the `HTTPServletRequest`, and these attributes should be accessed via the `getPipelineProperty` JSP tag (that is, the attributes exist only for the life of an HTTP request).

Basically, Pipeline Session and Request scoping differ by how long the attribute is retained. When an attribute is specified with the Request scope, it is available from the time it is set, up to and including the display of the next JSP. The attribute is automatically deleted when a new request starts. Therefore, Request scope is useful for temporary objects that will only be needed for one page. For example, search results from the product catalog are stored as Request-scoped attributes. Attributes that must live longer should be specified as Pipeline Session scope, which will cause them to be retained throughout the customer's session. If you know that a Pipeline session attribute is only required for the current request, use the Request scope.

Note: All attributes added to the Pipeline session should be serializable. If they are not, the server will generate an error when trying to serialize the Pipeline session, and thus no Pipelines will be executed. To assist in debugging, set the `pipelineSession.debug` property in the `weblogiccommerce.properties` file to `true`. Then, when a Pipeline session `setAttribute()` method is called, the server console will indicate whether the attribute is serializable or not.

Managing the Pipeline Session

It is important that the Pipeline and HTTP sessions are associated with each other and that they are updated in parallel. This section contains information about accessing and storing the Pipeline session that is important to maintaining this relationship.

Accessing the Pipeline Session

It is highly recommended that clients requiring access to the Pipeline session use one of the APIs of the `CommerceInputProcessor` base class. The `CommerceInputProcessor` class is responsible for creating a new Pipeline session (if required), and for associating the Pipeline session with the HTTP session.

Note: For more information about `CommerceInputProcessor`, see "The CommerceInputProcessor Base Class" on page 4-6.

Storing the Pipeline Session in the HTTP Session

Each time the Pipeline session is updated, the HTTP session also needs to be updated so that the Pipeline session is replicated across all the nodes in a cluster. Because the Webflow infrastructure is responsible for setting the Pipeline session to the HTTP session at appropriate times, it is highly recommended that none of the `InputProcessors` directly store the Pipeline session in the HTTP session.

Note: For more information about `InputProcessors`, see "Using the InputProcessor Interface" on page 4-5.

Extending Input Processors

In addition to using the input processors provided in the BEA WebLogic Commerce Server product, you can create your own input processors. This section describes the conventions you must follow when creating new input processors.

Note: It is expected that a Java/EJB programmer (or someone with similar technical knowledge and abilities) will develop new input processors.

Using the InputProcessor Interface

New input processors must implement the `InputProcessor` interface and must supply an implementation for the `process` method. The `process` method accepts an `HTTPServletRequest` object as a parameter and returns a string (such as success) if execution is successful, as shown in the following method signature:

```
public String process (HTTPServletRequest request) throws ProcessingException
```

Notes: For more information about the `InputProcessor` interface, see the *Javadoc*.

For information about how the `InputProcessor` interface can be used to handle session time outs, see "Handling Session Timeouts" on page 4-12.

Input Processor Exceptions

All input processors must throw the `ProcessingException` exception, or one of its subclasses. To obtain the `ProcessingException` exception's exception message, use the scriptlet shown in Listing 4-1.

Listing 4-1 Obtaining the ProcessingException Exception Message

```
<% String errorMsg =
(String)request.getAttribute(HttpRequestConstants.PIPELINE_MESSAGE); %>
```

Note: For more information about the `ProcessingException` exception, see the *Javadoc*.

The CommerceInputProcessor Base Class

`CommerceInputProcessor` is the abstract base class for all the BEA WebLogic Commerce Server classes that implement the `InputProcessor` interface. This class has a number of utility methods for all the derived classes to use, some of which allow you to:

- Get the `PipelineSession` object associated with the current session.
- Get the `PipelineSession` object only if the `HttpSession` is a valid session.

Note: For more information on the `CommerceInputProcessor` base class, see the *Javadoc*.

Input Processor Naming Conventions

The name of an input processor should end with the suffix `IP`. For example, an input processor that is responsible for deleting a shipping address might be called `DeleteShippingAddressIP`.

Input Processors and Statelessness

Because the Webflow controls the life cycle of input processors, the Webflow may create and destroy input processors without regard for the data that may be contained within them. Therefore, input processors should always be stateless, and it is recommended that you do not define any instance variables in an input processor.

Other Development Guidelines

Execution of business (application) logic should not be done within input processors. Specifically, input processors should not call EJBs or attempt to access a database. All such logic should be implemented in Pipeline components. Although it is possible to execute this logic within an input processor, doing so would defeat the purpose of the Webflow/Pipeline infrastructure and would not easily lend itself to modification.

By separating business logic from the presentation logic, your e-commerce site is inherently flexible in nature. Modifying or adding functionality can be as simple as creating and plugging in new Pipelines and/or input processors.

- For more information about input processors, see "Using Input Processors with Webflow" on page 2-10.
- For more information about Pipeline components, see "Using Pipelines with Webflow" on page 2-13.

Extending Pipelines and Pipeline Components

In addition to using the Pipelines and Pipeline components provided in the BEA WebLogic Commerce Server product, you can create your own Pipelines and Pipeline components. This section describes the conventions you must follow when creating new Pipelines and Pipeline components.

Note: It is expected that a Java/EJB programmer (or someone with similar technical knowledge and abilities) will develop new Pipelines and Pipeline components.

Using the PipelineComponent Interface

New Pipeline components must implement the `PipelineComponent` interface and must supply an implementation for the `process` method. The `process` method accepts a `PipelineSession` object as a parameter, and returns updated `PipelineSession` objects if the execution is successful, as shown in the following method signature:

```
public PipelineSession process(PipelineSession session) throws RemoteException,
PipelineNonFatalException, PipelineFatalException
```

Notes: For more information about the `PipelineComponent` interface, see the *Javadoc*.

For information about how the `PipelineComponent` interface can be used to handle session timeouts, see “Handling Session Timeouts” on page 4-12.

Pipeline Component Exceptions

Pipeline components may throw a `PipelineFatalException` to signify that the component has failed. When this occurs, no further Pipeline components are executed and if the Pipeline is transactional, the transaction will be rolled back.

To obtain the `PipelineFatalException` exception’s exception message, use the scriptlet shown in Listing 4-2.

Listing 4-2 Obtaining the PipelineFatalException Exception Message

```
<% String errorMsg =
(String)request.getAttribute(HttpRequestConstants.PIPELINE_MESSAGE); %>
```

Note: For more information about fatal Pipeline exceptions in a transactional Pipeline, see “Transactional Versus Nontransactional Pipelines” on page 4-11.

Pipeline components may also throw a `PipelineNonFatalException` to indicate that the component has failed, but that subsequent Pipeline components should be executed. Lastly, a Pipeline component may throw a `RemoteException`.

The Webflow integrates with these exceptions as follows:

- `PipelineFatalException`: If any component in a Pipeline throws a `PipelineFatalException` or a class derived from `PipelineFatalException`, besides aborting the Pipeline and the transaction, the Webflow will perform an exception search on the exception thrown.

Note: For a detailed description about how Webflow searches transitions, see "Webflow Search Order" on page 2-8.

- `RemoteException`: If the Pipeline throws a `RemoteException`, it is treated as a server error and the `servererror.jsp` is displayed.

When an exception search is performed, the Webflow looks for the exact exception found as the event. If this exception is not found, the Webflow will begin looking through the search order, as decribed in "Webflow Search Order" on page 2-8.

The CommercePipelineComponent Base Class

`CommercePipelineComponent` is an abstract base class for all the BEA WebLogic Commerce Server classes that implement the `PipelineComponent` interface. This class provides a utility method that allows you to obtain database connections from the `commercePool` (set up in the WebLogic Server console).

Note: For more information about the `CommercePipelineComponent` base class, see the *Javadoc*.

Pipeline Component Naming Conventions

The name of a Pipeline component should end with the suffix `PC`. For example, a Pipeline component that is responsible for saving a shopping cart might be called `SaveCartPC`.

Implementation of Pipeline Components as Stateless Session EJBs or Java Objects

Pipeline components can be implemented as either stateless session EJBs or as Java objects. Table 4-1 describes the differences between the two implementations.

Table 4-1 Comparison of Pipeline Component Implementations

Stateless Session EJBs	Java Objects
Heavier in weight and more complex to implement due to EJB overhead.	Lightweight, low overhead.
Server-provided instance caching.	No instance caching, possibly degrading performance.
Server-provided load balancing.	No load balancing, always executes on the node in the cluster where the Pipeline started execution.
Can use ACL-based security according to EJB specification.	Must manage security.

An implementing class that is a stateless session EJB must meet the following requirements:

- It must declare and implement a `create()` method in the bean's `Home` interface that takes no arguments and returns the appropriate `Remote` interface.
- It must declare and implement the `process()` method as part of its `Remote` interface.

Stateful Versus Stateless Pipeline Components

Whether Pipeline components are implemented as stateless session EJBs or as Java objects, Pipeline components themselves should be stateless. The business logic implemented in Pipeline components should only depend upon the `PipelineSession` object, the database, and other external resources. Should you define any instance variables, static variables, or static initializers within a Pipeline component, the results may be unpredictable.

Transactional Versus Nontransactional Pipelines

If all Pipeline components within the Pipeline will be invoked under one transaction, the respective Pipeline’s `isTransactional` property should be set to `true` in the Pipeline definition (within `pipeline.properties` file). Transactional Pipelines provide support for rolling back the database transaction and for making changes to the Pipeline session. If a transactional Pipeline fails, any database operations made by each of its Pipeline components are rolled back.

If a Pipeline component in a transactional Pipeline is implemented as a stateless session EJB, then its transaction attribute must be `Required`. Also, be sure that each of the Pipeline components in a transactional Pipeline has the correct transaction flag. Transaction flags indicate whether or not each bean will participate in the transaction. If the Pipeline’s `isTransactional` property is `true` and the participating Pipeline components (beans) have their transaction flag set to `never`, the Pipeline will fail to execute. Similarly, if the Pipeline’s `isTransactional` property is `false` and the Pipeline components have the transaction flag set to `mandatory`, the Pipeline will also fail to execute.

If a Pipeline component in a transactional Pipeline is implemented as a simple Java object, then for all database operations, the Pipeline component must use the Transactional DataSource associated with the connection pool, as defined in the WebLogic Server console. A transactional Pipeline containing Pipeline components implemented as simple Java objects commits the transaction upon success, and rolls back the transaction upon failure.

Other Development Guidelines

All server-side coding guidelines apply for development of new Pipeline components. Specifically:

- Avoid using threads.
- Avoid accessing the filesystem, since these operations are not thread-safe.

- Program all Pipeline components that are implemented as Java objects to be thread-safe.

Handling Session Timeouts

In any Web application, the `HttpSession` is usually short-lived. Therefore, every time the `HttpSession` is accessed, it must be evaluated to determine whether the session is new or whether the client has joined the current session. If the session is new and an attempt is made to access the `PipelineSession` from the `HttpSession`, then a null value will be returned unless it is recreated. This section describes in more detail how to handle session timeouts using the `InputProcessor` base classes.

Using the getPipelineSession() Method

The `CommerceInputProcessor` provides an overloaded `getPipelineSession()` method to help you handle session timeouts.

The first version of the `getPipelineSession()` method attempts to get the `PipelineSession` from the `HttpSession`. If the method is not able to locate the `PipelineSession`, then it will create a new instance and return a reference to the `PipelineSession`, as shown in the following method signature:

```
public PipelineSession getPipelineSession(HttpServletRequest request)
```

The second version of the `getPipelineSession()` method has an extra parameter, `checkValidity`, as shown in the following method signature. If `checkValidity` is `true` and the `HttpSession` is new, then the `getPipelineSession()` method throws an `InvalidSessionStateException` exception.

```
public PipelineSession getPipelineSession(HttpServletRequest
request, Boolean checkValidity) throws InvalidSessionStateException
```

Note: For more information about the `InvalidSessionStateException` exception, see “The InvalidSessionStateException Exception in webflow.properties” below.

The InvalidSessionStateException Exception in webflow.properties

The `InvalidSessionStateException` exception can be used for input processors. In the `webflow.properties` file, you can either provide the input processor name (as shown in the first line of Listing 4-3), or use the wildcard character (as shown in the second line of Listing 4-3).

Listing 4-3 Using InvalidSessionStateException in webflow.properties

```
InputprocessorName.inputprocessor.exception
(InvalidSessionStateException)= sessiontimeout.jsp

*.inputprocessor.exception(InvalidSessionStateException)=
sessiontimeout.jsp
```

The second option indicates that all input processors experiencing session timeout (throwing an `InvalidSessionStateException`) should load the `sessiontimeout.jsp` file.

Note: For more information about using the wildcard character in the `webflow.properties` file, see “Using the Wildcard Character” on page 2-5.

PipelineComponent and Session Timeouts

As part of handling session timeouts, each class that implements the `PipelineComponent` interface should determine whether or not a required attribute exists in the `PipelineSession` object. If the attribute does not exist, the subclass should throw an `InvalidPipelineSessionStateException` exception.

Note: For more information about the `InvalidPipelineSessionStateException` exception, see “The InvalidPipelineSessionStateException Exception in webflow.properites” below.

The InvalidPipelineSessionStateException Exception in webflow.properites

The `InvalidPipelineSessionStateException` exception can be used for Pipelines. In the `webflow.properties` file, you can either provide the Pipeline name (as shown in the first line of Listing 4-4), or use the wildcard character (as shown in the second line of Listing 4-4).

Listing 4-4 Using InvalidPipelineSessionStateException in webflow.properties

```
PipelineName.pipeline.exception
(InvalidPipelineSessionStateException)= sessiontimeout.jsp

*.pipeline.exception(InvalidPipelineSessionStateException)=
sessiontimeout.jsp
```

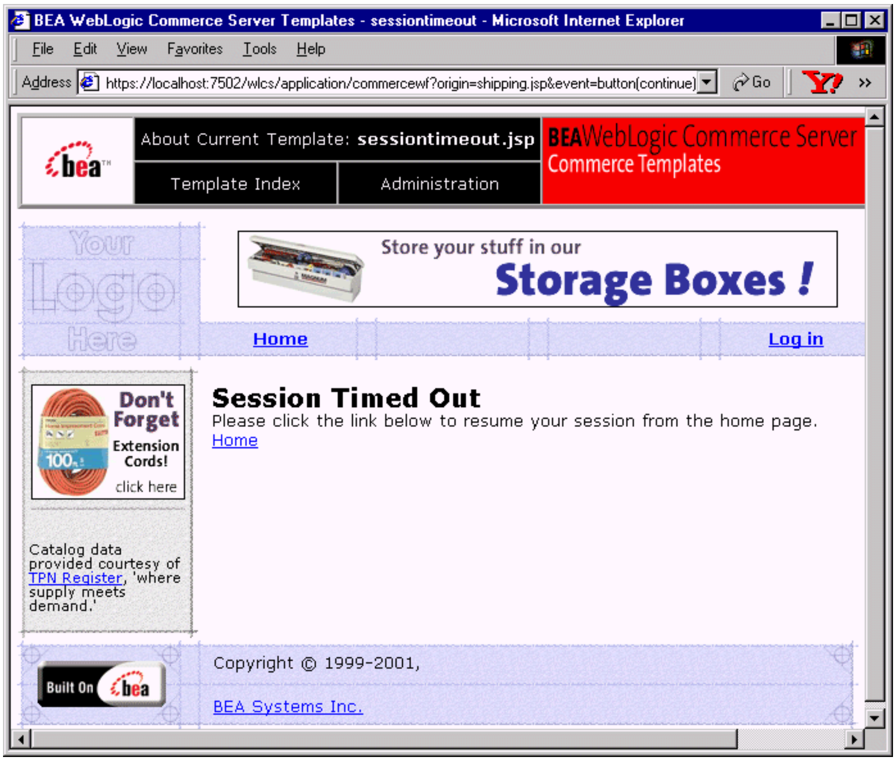
The second option indicates that all Pipelines experiencing session timeout (throwing an `InvalidPipelineSessionStateException`) should load the `sessiontimeout.jsp` file.

Note: For more information about using the wildcard character in the `webflow.properties` file, see “Using the Wildcard Character” on page 2-5.

About the sessiontimeout.jsp Template

The `sessiontimeout.jsp` template (shown in Figure 4-1) that ships with the BEA WebLogic Commerce Server product contains a link that calls the Webflow with the initial state, thereby giving the user a chance to start all over again.

Figure 4-1 The sessiontimeout.jsp Template


BEA WebLogic Commerce Server Templates - sessiontimeout - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address https://localhost:7502/wlcs/application/commercewf?origin=shipping.jsp&event=button(continue)
Go
About Current Template: sessiontimeout.jsp
Template Index
Administration
BEA WebLogic Commerce Server
Commerce Templates
Your Logo Here
Store your stuff in our Storage Boxes !
Home
Log in
Don't Forget Extension Cords! click here
Built On bea

Session Timed Out

Please click the link below to resume your session from the home page.

Home

Catalog data provided courtesy of TPN Register, 'where supply meets demand.'

Copyright © 1999-2001,

BEA Systems Inc.

CHAPTER

5 Webflow and Pipeline JSP Tags Library Reference

The BEA WebLogic Commerce Server product provides JSP tags specifically related to the Webflow and Pipeline mechanisms. This topic explains how to import each set of tags into your Web pages, and describes what each of these tags can do.

This topic includes the following sections:

- Webflow JSP Tags
  - <webflow:getValidatedValue>
  - <webflow:setValidatedValue>
- Pipeline JSP Tags
  - <pipeline:getPipelineProperty>
  - <pipeline:setPipelineProperty>

Webflow JSP Tags

The Webflow JSP tags are utility tags that simplify the implementation of JSPs that utilize the Webflow mechanism. To import the Webflow JSP tags, use the following code:

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

Note: For more information about the Webflow mechanism, see Chapter 1, "Overview of Webflow and Pipeline Management," in this guide.

Note: In the following tables, the Required column specifies if the attribute is required (yes) or optional (no). In the R/C column, C means that the attribute is a Compile time expression, and R means that the attribute can be either a Request time expression or a Compile time expression.

<webflow:getValidatedValue>

The `<webflow:getValidatedValue>` tag (Table 5-1) is used in a JSP to display the fields in a form that a customer must correct. The `<webflow:getValidatedValue>` tag is used in tandem with the`<webflow:setValidatedValue>` tag.

Table 5-1 <webflow:getValidatedValue>

Tag Attribute	Required	Type	Description	R/C
fieldName	Yes	String	The name of the field for which the status is desired.	R
fieldValue	Yes	String	The value as entered by the user.	R
fieldDefaultValue	No	String	The default value to use if the field value is missing.	R
fieldStatus	Yes	String	The status of the field. Valid values are: unspecified—Field was left blank; user must enter some data. invalid—User data is wrong. valid—User data is okay.	R
invalidColor	No	String	The color with which the label for an invalid field is to be marked. Defaults to red.	R
validColor	No	String	The color with which the label for a valid field is to be marked. Defaults to black.	R

Table 5-1 <webflow:getValidatedValue> (Continued)

Tag Attribute	Required	Type	Description	R/C
unspecifiedColor	No	String	If the user leaves a required field blank, this will be the color of the label for that field. Defaults to red.	R
fieldColor	No	String	The background color of the field.	R
fieldMessage	No	String	Specific message fo the current field.	R

These fields are determined and marked by an input processor after performing its validation activities. All `InputProcessors` use a `ValidatedValues` object to communicate which fields were successfully processed as well as those that were determined to be invalid.

Example 1

When used in a JSP, this sample code will obtain the current value and processing status of the `<field_name>` form field.

```
<webflow:getValidatedValue fieldName=”<field_name>”
fieldValue=”<field_value>” fieldStatus=”status” />
```

Example 2

The `<webflow:getValidatedValue>` tag refers to the webflow.tld tag library to retrieve available elements/attributes. In this example, a request is being made to obtain the following values from the HTTP session:

```
fieldName
fieldValue
fieldStatus
validColor
invalidColo
unspecifiedColor
fieldColor
```

These attributes are used for display purposes. (In this case, indicate that this field is required and mark it in red). The overall goal is to display values back to the user indicating which pieces are valid/invalid as returned from the input processor.

```
<webflow:getValidatedValue
fieldName="<%=HttpRequestConstants.CUSTOMER_FIRST_NAME%>"
fieldValue="customerFirstName" fieldStatus="status"
validColor="black"
invalidColor="red" unspecifiedColor="black" fieldColor="fontColor"
/>
```

<webflow:setValidatedValue>

The `<webflow:setValidatedValue>` tag (Table 5-2) is used in a JSP to configure the display of fields in a form that a customer must correct. Usually this is done within an Inputprocessor, but it can also be done from a JSP by using this tag. The `<webflow:setValidatedValue>` is used in tandem with the `<webflow:getValidatedValue>` tag.

Table 5-2 <webflow:setValidatedValue>

Tag Attribute	Required	Type	Description	R/C
fieldName	Yes	String	The name of the field for which the status is desired.	C
fieldValue	Yes	String	The new value of the field.	C
fieldStatus	No	String	The processing status of the field. Valid values are: unspecified—Field was left blank; user must enter some data. invalid—User data is wrong. valid—User data is okay.	C

Example

When used in a JSP, this sample code will obtain the current value and processing status of the `<field_name>` form field.

```
<webflow:setValidatedValue fieldName=”<field_name>”
fieldValue=”<field_value>” fieldStatus=”status” />
```

About the ValidatedValues Java Class

The `ValidatedValues` class allows a Java/EJB programmer who writes an `InputProcessor` to report the status of processed form fields back to the commerce engineer/JSP content developer.

The constructor for the `ValidatedValues` class takes an `HTTPServletRequest` as a parameter, as shown in the following method signature:

```
public ValidatedValues (javax.servlet.http.HttpServletRequest s)
```

The public methods used to convey the status of the validation through the `getValidatedValue` and `setValidatedValue` JSP tags are shown in Table 5-3.

Table 5-3 ValidatedValues Public Methods

Method Signature	Description
`public String getStatus (String name)`	Retrieves the status for the specified field, which may be unspecified, invalid, or valid.
`public void setStatus (String name, String value)`	Sets the status for the specified field.
`public String getValue (String name)`	Retrieves the current value for the specified field.
`public void setValue (String name, String value)`	Sets the value for the specified field.

Pipeline JSP Tags

The Pipeline JSP tags are used to store and retrieve attributes in a Pipeline session. To import the Pipeline JSP tags, use the following code:

```
<%@ taglib uri=“pipeline.tld” prefix=“pipeline” %>
```

Note: For more information about the Webflow mechanism, see Chapter 1, “Overview of Webflow and Pipeline Management,” in this guide.

Note: In the following tables, the Required column specifies if the attribute is required (yes) or optional (no). In the R/C column, C means that the attribute is a Compile time expression, and R means that the attribute can be either a Request time expression or a Compile time expression.

<pipeline:getPipelineProperty>

The `<pipeline:getPipelineProperty>` tag (Table 5-4) retrieves a named attribute (property) from the Pipeline session object, from a property of one of the objects that has been retrieved from the Pipeline session, or from the request. Objects that are stored as attributes of the Pipeline session must conform to the standard bean interface and implement a `get<Attribute>()` method for each publicly accessible attribute. If `propertyName` does not exist in the Pipeline session, then `returnName` contains `null`.

Table 5-4 <pipeline:getPipelineProperty>

Tag Attribute	Required	Type	Description	R/C
propertyName	No	String	The key in the PipelineSession hash table that identifies what property value to get. If omitted, the PipelineSession itself is returned.	R
pipelineObject	No	Object	Name of the object in which to search for the key specified as `propertyName`. If no object is specified, the default is to look in Pipeline session.	R

Table 5-4 <pipeline:getPipelineProperty> (Continued)

Tag Attribute	Required	Type	Description	R/C
returnName	No	String	Name of the variable that will contain the value of the attribute. If omitted, the tag is created inline (that is, the variable is not returned but the `toString()` method is called and the results are displayed to the browser).	R
returnType	No	String	A valid type for the returned attribute. If this is not specified, the value is returned as an object.	R

attributeScope	No	Int	The scope in which to look for the key specified as `propertyName`.
			Valid values are:
			`PipelineConstants.SESSION_SCOPE`
			`PipelineConstants.REQUEST_SCOPE`
				R

Example

When used in a JSP, this sample code will retrieve an attribute named `<property_name>` and store it in a variable named `<return_name>`. The type of this variable will be `<return_type>`, and the scope to which this attribute belongs is specified by `<attribute_scope>`.

```
<pipeline:getPipelineProperty propertyName=”<property_name>”
returnName=”<return_name>” returnType=”<return_type>”
attributeScope=”<%=<attribute_scope>%>”/>
```

Note: For more information about the scope of Pipeline session attributes, see the section “Attribute Scoping” on page 4-3 in this guide.

<pipeline:setPipelineProperty>

The `<pipeline:setPipelineProperty>` tag (Table 5-5) sets a named attribute (property) to the Pipeline session object or to a property of one of the objects that has been retrieved from the Pipeline session. Objects that are stored as attributes of the

Pipeline session must conform to the standard bean interface and implement a `set<propertyName>()` method for each publicly accessible attribute. Scope defaults to `PipelineConstants.SESSION_SCOPE`.

Table 5-5 <pipeline:setPipelineProperty>

Tag Attribute	Required	Type	Description	R/C
propertyName	Yes	String	Name of the key with which the given property is to be associated.	R
propertyValue	Yes	Object	The value to associate with the `propertyName`.	R
pipelineObject	No	Object	Name of the object in the Pipeline session in which to store the given key and value pair.	R

If `pipelineObject` is not specified, then the given property and its value will be set to the Pipeline session. If the `pipelineObject` is specified, then the object must implement the `set<PropertyName>()` method, which takes two parameters: a property name (`String`) and a property value (`Object`), as shown in the following method signature:

```
public void set<PropertyName>(String propertyName,java.lang.Object
propertyValue);
```

Note: If the `set<PropertyName>()`method is not implemented, an exception will be thrown during the processing of the JSP that has the `setPipelineProperty` tag in it.

Example

When used in a JSP, this sample code will set the property named `<property_name>` of the `<pipeline_object_name>` with the value specified in `<property_value>`.

```
<pipeline:setPipelineProperty propertyName=”<property_name>”
propertyValue=”<property_value>”
pipelineObject=”<pipeline_object_name>”/>
```

Note: The `<pipeline_object_name>` must be a fully qualified class name.

Index