



# BEA WebLogic Application Integration

A Component of BEA WebLogic Integration

## User Guide

BEA WebLogic Application Integration Release 2.0  
Document Edition 2.0  
July 2001

## Copyright

Copyright © 2001 BEA Systems, Inc. All Rights Reserved.

## Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

## Trademarks or Service Marks

BEA, WebLogic, Tuxedo, and Jolt are registered trademarks of BEA Systems, Inc. How Business Becomes E-Business, Operating System for the Internet, Liquid Data, BEA WebLogic E-Business Platform, BEA Builder, BEA Manager, BEA eLink, BEA Campaign Manager for WebLogic, BEA WebLogic Commerce Server, BEA WebLogic Personalization Server, BEA WebLogic Process Integrator, BEA WebLogic Collaborate, BEA WebLogic Enterprise, BEA WebLogic Server, and BEA WebLogic Integration are trademarks of BEA Systems, Inc.

All other product names may be trademarks of the respective companies with which they are associated.

### **BEA WebLogic Application Integration**

<b>Document Edition</b>	<b>Part Number</b>	<b>Date</b>	<b>Software Version</b>
2.0	N/A	July 2001	2.0

---

# Contents

## About This Document

What You Need to Know .....	vii
e-docs Web Site .....	viii
How to Print the Document .....	viii
Related Information .....	viii
Contact Us! .....	ix
Documentation Conventions .....	x

## 1. Introduction to Using Application Integration

Before You Begin .....	1-2
Concepts .....	1-2
Adapters: XML Interfaces to an EIS .....	1-3
Application Views: Windows to an EIS Application .....	1-3
When to Use Application Views and	
When to Write Custom Code .....	1-4
When to Define Application Views .....	1-4
When to Write Custom Code Instead of	
Defining Application Views .....	1-4
“Defining” Versus “Using” Application Views .....	1-5
Defining: Configuring the Application View	
and Adding Events and Services .....	1-5
Using Application Views in Business Processes .....	1-5
Services and Events: Specialized Message Handlers .....	1-6
Understanding Services .....	1-6
Understanding Events .....	1-7
Using the Application View Management Console .....	1-7
Who Uses Adapters and Application Views? .....	1-8

---

System Administrators .....	1-8
Adapter Developers .....	1-8
Adapter Users .....	1-8
How Responsibilities are Distributed Among Users.....	1-8
Defining Application Views.....	1-9
Creating and Configuring an Application View .....	1-10
Adding Services and Events to an Application View .....	1-10
Testing Services and Events.....	1-10
Using Application Views in Business Processes.....	1-10
Using Application Views in WebLogic Process Integrator .....	1-11
Using Application Views by Writing Custom Code .....	1-11
Deciding Which of the Two Methods to Use.....	1-11

## 2. Defining Application Views

Before You Begin .....	2-2
Introduction to Defining Application Views .....	2-2
The Flow of Events .....	2-2
Steps for Defining Application Views.....	2-4
Logging On to the Application View Management Console .....	2-5
Defining an Application View.....	2-6
Adding Services to an Application View .....	2-9
Adding Events to an Application View.....	2-11
Deploying an Application View .....	2-13
Undeploying an Application View .....	2-18
Testing an Application View's Services .....	2-19
Testing an Application View's Events .....	2-23
If you choose Service .....	2-24
If you choose Manual .....	2-27
Editing an Application View .....	2-30

---

### 3. Using Application Views in WebLogic Process Integrator

Before You Begin.....	3-2
Introduction to Using Application Views in WebLogic Process Integrator .....	3-3
Using Application Views in WebLogic Process Integrator .....	3-3
Scenario 1: Setting Up a Task Node to Call an Application View Service .....	3-4
Steps for Setting up a Task Node to Call an Application View Service .....	3-4
Scenario 2: Setting Up an Event Node to Wait for a Response from an Asynchronous Application View Service .....	3-10
Receiving an Asynchronous Application View Service Response .....	3-11
Handling Errors in an Asynchronous Application View Service Response .....	3-11
Steps for Setting Up an Event Node to Wait for a Response from an Asynchronous Application View Service .....	3-11
Explanation of Functions Provided by the AI Plug-in .....	3-14
Scenario 3: Creating a Workflow that is Started by an Application View Event .....	3-16
Steps for Creating a Workflow that is Started by an Application View Event .....	3-16
Scenario 4: Setting Up an Event Node to Wait for an Application View Event .....	3-19
Steps for Setting Up a Node to Wait for an Application View Event .....	3-19

### 4. Using Application Views by Writing Custom Code

Before You Begin.....	4-2
Introduction to Using Application Views by Writing Custom Code .....	4-2
Steps for Using Application Views by Writing Custom Code.....	4-3
About this Example .....	4-3
Prerequisites for this Example.....	4-3
Writing the Java Class .....	4-4
Example Code for SyncCustomerInformation .....	4-6

---

## 5. Using the Application View Management Console

Before You Begin .....	5-2
Introduction to Using the Application View Management Console .....	5-2
Steps for Using the Application View Management Console .....	5-2
Logging On to the Application View Management Console .....	5-3
Creating Folders .....	5-4
Removing Application Views .....	5-5
Removing Folders .....	5-5

---

# About This Document

The *BEA WebLogic Application Integration User Guide* is organized as follows:

- “Introduction to Using Application Integration” provides an overview of BEA WebLogic Integration Framework and explains how it fits into the WebLogic Server environment and contributes to the BEA EAI solution.
- “Defining Application Views” explains how to log into an adapter, create and configure application views to represent your enterprise’s business processes.
- “Using Application Views in WebLogic Process Integrator” explains how to use application views in the WebLogic environment by setting up workflows using WebLogic Process Integrator.
- “Using Application Views by Writing Custom Code” explains how to use application views in the WebLogic environment by writing custom Java code.
- “Using the Application View Management Console” explains how to use namespaces to organize your application views by location or department instead of by adapter.

## What You Need to Know

This document is intended for the following users:

- **Business Analysts**—Business analysts work with the technical analysts to ensure accuracy of the business interface functionality, to create application views, and to use application views within your enterprise.
- **Technical Analysts**—Technical analysts are responsible for configuring an adapter, for setting up WebLogic services to execute information transfers with a

---

legacy system, for configuring solutions using adapters, and for evaluating, mapping, deploying, and maintaining the WebLogic Environment. This guide assumes that the technical analyst has thorough knowledge of the entire system.

## e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation or go directly to the “edocs” Product Documentation page at <http://edocs.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 BEA WebLogic Application Integration documentation home page on the edocs Web site. You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the BEA WebLogic Application Integration 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 resources are also available:

- BEA WebLogic Server documentation (<http://e-docs.bea.com>)



- 
- BEA WebLogic Process Integrator documentation (<http://www.edocs.com>)
  - XML Schema Specification (<http://www.w3.org/TR/xmlschema-1/>)
  - The Sun Microsystems, Inc. Java site (<http://www.javasoft.com/>)
  - The Sun Microsystems, Inc. J2EE Connector Architecture Specification (<http://java.sun.com/j2ee/connector/>)

## Contact Us!

Your feedback on the BEA WebLogic Application Integration 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 BEA WebLogic Application Integration documentation.

In your e-mail message, please indicate that you are using the documentation for the BEA WebLogic Application Integration 2.0 release.

If you have any questions about this version of BEA WebLogic Application Integration, or if you have problems installing and running BEA WebLogic Application Integration, 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
<b>boldface text</b>	Indicates terms defined in the glossary.
<b>Ctrl+Tab</b>	Indicates that you must press two or more keys simultaneously.
<i>italics</i>	Indicates emphasis or book titles.
monospace text	Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard. <i>Examples:</i> chmod u+w * c:\startServer .doc wls.doc BITMAP float
<b>monospace boldface text</b>	Identifies significant words in code. <i>Example:</i> void <b>commit</b> ( )
<i>monospace italic text</i>	Identifies variables in code. <i>Example:</i> String <i>expr</i>
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. <i>Examples:</i> LPT1 SIGNON OR

---

Convention	Item
...	Indicates one of the following in a command line: <ul style="list-style-type: none"> <li>■ That an argument can be repeated several times in a command line</li> <li>■ That the statement omits additional optional arguments</li> <li>■ That you can enter additional parameters, values, or other information</li> </ul> <i>Example:</i> <pre>import com.sap.rfc.exception.*;</pre>
.	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

---



# 1 Introduction to Using Application Integration

Welcome to the *BEA WebLogic Application Integration User Guide*, the guide for using adapters built using the BEA Application Integration ADK (Adapter Development Kit). This document explains how to define application view services and events and use them in your business processes.

The instructions in this document are general instructions. Because each adapter and application is different, the instructions do not cover any information specific to any particular adapter or application. If you are looking for a tour of specific adapters developed using Application Integration ADK, see the following two sections in “BEA WebLogic Application Integration Development Guide:”

- E-mail Adapter Guide
- DBMS Adapter Guide

This section provides information on the following subjects:

- Before You Begin
- Concepts
  - Adapters: XML Interfaces to an EIS
  - Application Views
  - When to Use Application Views and When to Write Custom Code
  - “Defining” Versus “Using” Application Views
  - Services and Events: Specialized Message Handlers
  - Using the Application View Management Console

# 1 Introduction to Using Application Integration

---

- Who Uses Adapters and Application Views?
- Defining Application Views
- Using Application Views in Business Processes

## Before You Begin

Before you can begin using adapters to integrate your enterprise, make sure the following prerequisites are satisfied:

- You have installed WebLogic 6.0, with Service Pack 2.
- You have installed JDK 1.3. The JDK 1.3 development kit is automatically installed when you install WebLogic 6.0, although you may want to install your own version, as long as it is 1.3-compliant.
- You have installed the Application Integration Plug-in for WebLogic Process Integrator.
- Deploy each WebLogic adapter for which you will define application views.

**Note:** For a complete list of prerequisites, see *BEA WebLogic Application Integration Release Notes*.

## Concepts

This section describes important concepts with which you should familiarize yourself before you use Application Integration. The following concepts are discussed in detail in “Defining Application Views” and “Using Application Views in WebLogic Process Integrator.” For a broad overview of BEA WebLogic Application Integration, see the *BEA WebLogic Integration Framework Getting Started Guide*.

---

## Adapters: XML Interfaces to an EIS

In Application Integration, an adapter is software that forms an XML-based interface for connecting an Enterprise Information Server (EIS) and your enterprise's WebLogic server. An adapter enables application-level communication between the WebLogic server and a particular EIS in your enterprise. In your organization, you probably have a variety of separate EIS systems whose applications must interact with each another. In the Application Integration environment, you don't have to connect each EIS to every other EIS in your enterprise. Instead, you develop an adapter for each EIS.

## Application Views

The application view is the cornerstone of BEA WebLogic Application Integration's Integration Framework. It provides a view of the application capabilities exposed by an adapter that a user can customize to meet specific business needs. A user tailors an application view for a specific business purpose, and as a result, the application view provides an effective alternative to the "one size fits all" approach that most applications provide for the design of their client interface. The application view allows you to define for it only the business capabilities that directly apply to your business purpose. You can customize the capabilities by naming, describing and defining their data requirements. For more information about application views, see the *BEA WebLogic Application Integration Adapter Development Guide*.

For each adapter installed on your WebLogic server, you can define any number of application views. An application view represents a subset of business functions available on the adapter's EIS. An application view accepts requests for service invocation from WebLogic clients and invokes the proper system functions on the target EIS. To communicate with the EIS, an application view makes use of connections provided by its adapter.

An application view has the following basic properties:

- Communication parameters
- Services
- Events

# 1 Introduction to Using Application Integration

---

An application view's communication parameters provide the "rules of the road" used by the WebLogic server and the EIS application to communicate and perform housekeeping. An application view is a business-level interface that uses the capabilities of the EIS, via the adapter, to implement the business function of a service and to deliver events of significance to the business. You can add any number of services and events to each application view. For more information on events and services, see the *BEA WebLogic Application Integration Adapter Development Guide*. For more information on defining application views, see "Introduction to Defining Application Views."

## When to Use Application Views and When to Write Custom Code

To support service invocation and events, you can define application views, or you can write custom code that accomplishes the same functions. Application views provide the most convenient interface to an adapter's resources, but there are other ways to access an adapter. Normally, for each WebLogic adapter, you will define application views to expose the application functions. However, for those who require more control, you may also write custom code to access the resources of an adapter. For your enterprise, you must decide whether to define application views or write your own code.

### When to Define Application Views

You can define application views to easily integrate most EIS applications. In general, define application views in the following situations:

- You have more than one EIS system in your enterprise, and you lack developers who have detailed, thorough knowledge of all of the systems.
- You want to use WebLogic Process Integrator to construct business processes.
- When you may need to update the parameters of the adapter or one of its processes.



---

## When to Write Custom Code Instead of Defining Application Views

In general, write custom code as an interface to an adapter only in the following situations:

- When you have only one EIS system in your enterprise
- When you have access to a developer who has thorough, detailed knowledge of each EIS involved in the business processes being coded.
- When you do not need to use the coded functions in WebLogic Process Integrator.
- When your code will never change.

## “Defining” Versus “Using” Application Views

There are two initial steps in the life cycle of an application view:

- Defining the application view.
- Using the application view.

### Defining: Configuring the Application View and Adding Events and Services

When you define an application view, you configure the communication parameters, then add services and/or events. The application view’s services and events expose specific functions of the application. The communication parameters of the application view govern how the application view will connect to the target EIS.

Defining an application view includes the following tasks:

- Entering a unique name for the application view.
- Configuring parameters that establish the network connection between the application view and the application itself.
- Configuring parameters specific to the application.

# 1 Introduction to Using Application Integration

---

- Configuring parameters used for load balancing by the application view.
- Configuring parameters used to manage the pool of connections available to the application view.
- Defining security privileges for users of the application view.

## Using Application Views in Business Processes

After you define an application view, you can deploy it on the WebLogic server. You can use deployed application views to implement your enterprise's business processes.

After using an application view in a workflow, the end result is a deployed electronic representation of your enterprise's business process. The workflow specifies how your enterprise's EIS applications will interact with each other to accomplish the business processes. The application views perform the transactions themselves.

## Services and Events: Specialized Message Handlers

For each application view, you can add any number of services and events, which support specific types of transactions between the WebLogic server and the target EIS application. Services and events provide an XML-based interface between the WebLogic server and the functions of the target EIS application.

When you add services and events to an existing application view, all you need to know are the application-level parameters for the transaction. Each adapter is programmed to perform the services and events in a way that is appropriate for your WebLogic network.

## Understanding Services

For each application view, you can designate any number of services. Services are self-describing software modules that expose a simple XML-based request/response interface. An application view's services receive a particular type of XML document from the client and then perform a specified transaction on a designated EIS application. When the EIS application receives the message from the service, it may or may not send back a response to the service. Each service represents a particular transaction the EIS application can perform, and the application view manages the mapping of its services to EIS functions.

---

## Synchronous and Asynchronous Services

You must designate each service to be either synchronous or asynchronous.

- **Asynchronous services:** When an EIS application transmits a service request to an asynchronous service, the application continues its processing without waiting for an immediate response from the service. Use asynchronous services when the application does not need the service response until a later stage in its business process.

For example, assume one of your application views has a service called `CreateCustomer`. Whenever an entity sends a customer record to `CreateCustomer`, `CreateCustomer` sends a message to its corresponding EIS application, known as `CustomerRelationshipApplication`. When `CustomerRelationshipApplication` receives the message from `CreateCustomer`, the application writes a new record to its customer database, using the data supplied by the entity that called `CreateCustomer`. Because the `CustomerRelationshipApplication` is programmed not to respond with an acknowledgement, you must designate the service to be asynchronous.

- **Synchronous services:** When an EIS application transmits a service request to a synchronous service, the application waits for a response from the service before the application continues processing. Use synchronous services when the application's business process requires an immediate service response before it can continue.

For example, assume one of your application views has a service called `QueryCustomer`. Whenever an entity calls `QueryCustomer`, `QueryCustomer` sends a message to its corresponding EIS application, known as `CustomerRelationshipApplication`. When `CustomerRelationshipApplication` receives the message from `QueryCustomer`, the application retrieves customers or a list of customers requested by the calling entity. According to your enterprise's business processes, the calling entity must receive an acknowledgement to notify it that the transaction was successful. In this case, you must designate the service to be synchronous.

For each service, the type you choose depends on the EIS application. For more information on the EIS, contact the appropriate technical analyst.

## Understanding Events

For each application view, you can designate any number of events. An application view event responds to application states, extracts data about the event from the EIS, then propagates the data into the WebLogic environment.

## Using the Application View Management Console

You can organize all of your application views into folders so you can look them up using your own organizational scheme. Once you set up a folder tree, you can browse all application views in your enterprise without regard to the adapter used to define the application view. You can use the Application View Management Console to create new folders and to define new application views in them. For details, see “Using the Application View Management Console.”

## Who Uses Adapters and Application Views?

In your enterprise, several people may share the responsibilities of maintaining WebLogic adapters, application views, and their services and events. In most enterprises, these people can be grouped into one of three categories:

- System administrators
- Adapter developers
- Adapter users

### System Administrators

If you are the one responsible for installing the BEA WebLogic Application Integration ADK, then you are the system administrator referred to throughout this document.

## Adapter Developers

If you are a software developer or a high-level technician in your enterprise, you are probably an adapter developer. Adapter developers commonly use the ADK to develop new adapters for EIS systems and design the accompanying user interface that the adapter user will interact with when creating application views for the adapter. For information on developing adapters, see the *BEA WebLogic Application Integration Adapter Development Guide*.

## Adapter Users

If you are a business analyst, EIS specialist, or technical analyst in your enterprise, you are probably an adapter user. Adapter users do not usually develop adapters but may provide specifications to the adapter developer. Once an adapter is developed, adapter users normally define and manage its application views.

## How Responsibilities are Distributed Among Users

Table 1-1 shows how tasks and responsibilities are typically divided.

# 1 Introduction to Using Application Integration

---

**Table 1-1 Common Jobs and Their Owners**

<b>Task</b>	<b>System Administrator</b>	<b>Adapter Developer</b>	<b>Adapter User</b>
Installing the Application Integration	X		
Developing an adapter		X	
Developing the user interface used for defining application views		X	
Configuring Application Integration to deploy an adapter	X	X	
Defining an application view			X
Configuring connection parameters of application views			X
Adding and testing services and events			X
Setting up application view folders			X

## Defining Application Views

When you define an application view for an adapter, you are creating an XML-based interface between WebLogic and a particular EIS application. For detailed steps for defining application views for adapters, see “Defining Application Views.”

Defining an application view involves these basic steps:

- Creating and Configuring and Application View.
- Adding Services and Events to an Application View.
- Testing Services and Events.

---

## Creating and Configuring an Application View

The first step in defining an application view for an adapter is to log on to the Application View Administration Console, choose a folder where the application view will reside, and configure its communication parameters. For details on creating and configuring an application view, see the following topics:

- “Logging On to the Application View Management Console.”
- “Defining an Application View.”

## Adding Services and Events to an Application View

After defining the communication parameters of the application view, the next step is to add services and events. Services and events support a subset of an application’s business processes by allowing other WebLogic entities to interact with the application functions you specify. The application view services and events allow specific types of transactions between the WebLogic server and the EIS application. For details on adding services and events to an application view, see the following topics:

- “Adding Services to an Application View.”
- “Testing an Application View’s Events.”

## Testing Services and Events

After adding a service or event to an application view, you must make sure the service or event interacts properly with the EIS application. For details on testing services and events, see the following topics:

- “Testing an Application View’s Services.”
- “Testing an Application View’s Events.”

## Using Application Views in Business Processes

Once you define an application view in your WebLogic environment, you can use the application view in your enterprise’s business processes. There are two ways to use application views in business processes:

# 1 Introduction to Using Application Integration

---

- By designing workflows in WebLogic Process Integrator.
- By writing custom Java code.

## Using Application Views in WebLogic Process Integrator

The most common way to use an application view in your enterprise's business processes is to design a workflow in WebLogic Process Integrator. Process Integrator provides a GUI-based environment for designing business process workflows. These workflows can include application view services and events defined using Application Integration.

There are four ways to use an application view in a workflow using WebLogic Process Integrator:

- Scenario 1: Setting Up a Task Node to Call an Application View Service
- Scenario 2: Setting Up an Event Node to Wait for a Response from an Asynchronous Application View Service
- Scenario 3: Creating a Workflow that is Started by an Application View Event
- Scenario 4: Setting Up an Event Node to Wait for an Application View Event

For detailed information on each method, see "Using Application Views in WebLogic Process Integrator."

## Using Application Views by Writing Custom Code

If you do not use WebLogic Process Integrator, the alternate way to use an application view in your enterprise is to write custom Java code to implement a business process.

For detailed steps for custom coding business processes, see "Using Application Views by Writing Custom Code."

## Deciding Which of the Two Methods to Use

For each business process you implement, you will need to decide which of the two implementation methods to use. You can implement any business processes as a workflow using WebLogic Process Integrator, but you should only attempt to custom code a business processes if it is extremely simple and specialized. In this document, custom coding is offered only as an alternate method for those who require it.



---

## When to Use WebLogic Process Integrator

In general, use WebLogic Process Integrator to implement a business process in the following situations:

- When implementing the required business processes would require complicated error management, persistent processes, and sophisticated conditional branching.  
For example, if a business process receives events, selects only a subset of the events, performs complex branched actions, then generates many complex messages and sends the messages to a variety of WebLogic clients, then you should use WebLogic Process Integrator to implement the business process.
- When you will have to make occasional changes to the business process.  
WebLogic Process Integrator reduces the number of compile/test/debug cycles.
- When, like most organizations, your developers are valuable and scarce.

## When to Write Custom Java Code

In general, write custom code to implement a business process only in the following situations:

- When the business process is simple, without complicated error recovery, long-lived processes, conditional branching, or joining of the process flow.  
For example, if a business process performs a limited set of actions on an incoming message, then routes the minimally transformed message to a small number of client applications, then the business process is simple enough to express by writing custom code.
- When you will not need to update the business process very often.  
When you update custom code, the change requires a full compile/test/debug cycle, which can be costly.
- When your organization can afford to dedicate developers to implement the business processes in code.

# **1** Introduction to Using Application Integration

---

# 2 Defining Application Views

This section contains information on the following subjects:

- Before You Begin
- Introduction to Defining Application Views
  - The Flow of Events
- Steps for Defining Application Views
  - Logging On to the Application View Management Console
  - Defining an Application View
  - Adding Services to an Application View
  - Adding Events to an Application View
  - Testing an Application View's Services
  - Testing an Application View's Events
  - Editing an Application View

# Before You Begin

Before you attempt to define an application view, make sure the following prerequisites are satisfied.

- The appropriate WebLogic adapter has been developed. You can only create and configure application views for existing WebLogic adapters.
- Determine which business processes need to be supported by the application view you are configuring. The required business processes determine the types of services and events you will include in the adapter's application views. Normally, this means gathering information about the application's business requirements from the business analyst. Once you determine the necessary business processes, you can define and test the appropriate services and events.

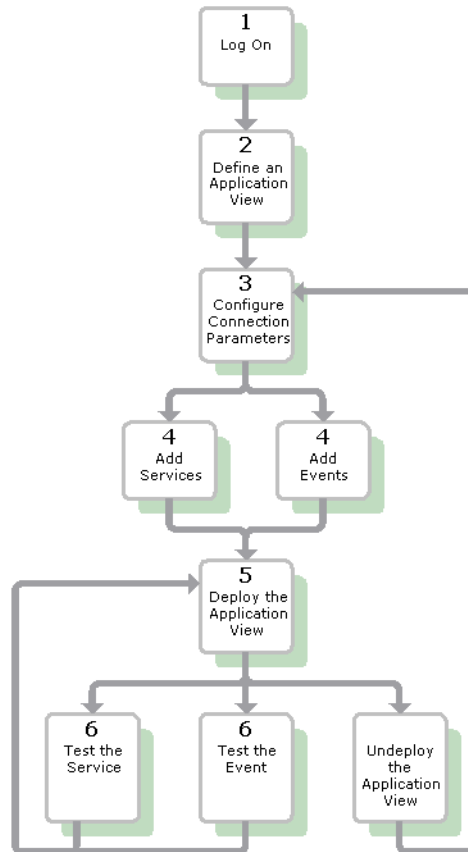
# Introduction to Defining Application Views

When you define an application view, you are creating an XML-based interface between your WebLogic server and a particular EIS application within your enterprise. Once you create the application view, a business analyst can use it to create business processes that use the application. For any adapter, you can create any number of application views, each with any number of services and events.

## The Flow of Events

Figure 2-1 shows an overview of the steps involved in defining an application view.

Figure 2-1 The Flow of Events for Defining and Configuring Application Views



1. Log on to the Application View Management Console. For detailed information, see “Logging On to the Application View Management Console.”
2. Click Add Application View to create a new application view for the appropriate WebLogic adapter. An application view enables a set of business processes for this adapter’s target EIS application. For detailed information, see “Defining an Application View.”
3. At the Configure Connection Parameters page, enter application connection parameters. For detailed information, see “Defining an Application View.”

## 2 Defining Application Views

---

The information is validated, and the application view is configured to connect to the system you specified.

4. Click Add Event or Add Service to define the appropriate events and services for this application view.
5. Deploy the application view on the WebLogic server so other entities can interact with it according to your security settings.

**Note:** You can only test an application view if it is deployed.

6. Test the services and events to make sure they can properly interact with the target EIS application.

Once the services and events are tested and functioning, you can use the application view in workflows. For more information, see “Using Application Views in WebLogic Process Integrator.”

7. Undeploy the application view if you need to reconfigure its connection parameters or add services and events.

**Note:** When an application view is undeployed, no other entities can interact with it.

## Steps for Defining Application Views

This section explains how to define and maintain application views using an EIS adapter for a fictional database EIS called simply “DBMS.” When you create application views for your enterprise, they may look different than the screens shown in this document. This is normal, because the application view’s adapter determines the information required for each application view page, and each enterprise has its own specialized adapters. For details on a WebLogic adapter used in your enterprise, consult the relevant technical analyst or EIS specialist.

## Logging On to the Application View Management Console

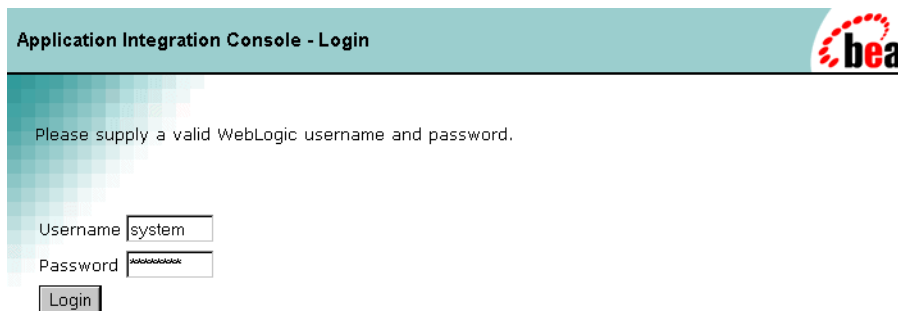
The first step in creating a new application view is to log on to the Application View Management Console page. The Application View Management Console displays all the application views in your WebLogic environment, organized into folders.

To log on to the Application View Management Console:

1. Open a new browser window.
2. Open the URL for your system's Application View Management Console. The actual URL you enter depends on your system. It should follow the format:

```
http://localhost:7001/wlai
```

The Application Integration Console - Login page displays.



Application Integration Console - Login

Please supply a valid WebLogic username and password.

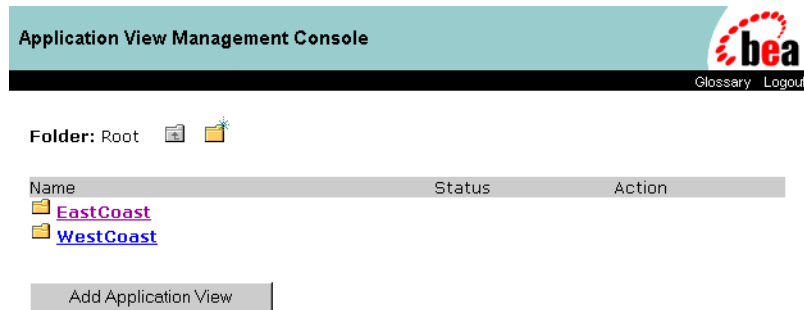
Username

Password

3. To log on to the Application View Management Console, enter your WebLogic username and password, then click Login. The Application View Management Console displays.

## 2 Defining Application Views

---



**Notes:** If you do not see a page like this, consult the WebLogic system administrator.

To add a folder, click the New Folder icon. For more information, see “Creating Folders.”


### Defining an Application View

After you log on to the Application View Management Console and navigate to a folder, click Add Application View to define an application view. Defining an application view exposes a subset of the functions of an EIS application.

1. Log on to the Application View Management Console (see “Logging On to the Application View Management Console.”).
2. To add a new application view to the current folder, click Add Application View. The Define New Application View page displays.



### Define New Application View

[Glossary](#) [Logout](#)

This page allows you to define a new application view in the EastCoast.Sales folder. If you indicate which adapter the application view is associated with, then you will be forwarded to the adapter specific design-time Web application.

Folder [EastCoast.Sales](#)

Application View Name\*

Description

Associated Adapters

**Note:** Once you define the application view, you can not move it to another folder.

3. In the Application View Name field, enter a name. The name should describe the set of functions performed by this application. Each application view name must be unique to its adapter. Valid characters are a–z, A–Z, 0–9, and \_ (underscore).
4. In the Description field, enter any relevant notes. These notes are viewed by users when they use this application view in workflows using WebLogic Process Integrator.
5. From the Associated Adapters list, choose the WebLogic adapter to use to create this application view.
6. Click OK. The Configure Connection Parameters page displays.

## 2 Defining Application Views

**Configure Connection Parameters**

Adapter Home WLIF Home Page WebLogic Console Glossary Logout

**Configure Connection** *On this page, you supply parameters to connect to your DBMS*

Administration  
Add Service  
Add Event  
Deploy Application View

WebLogic User Name\* system

WebLogic Password\* [masked]

XA Compliant Data Source?

Data Source Name (JNDI)\* WLAJ\_DataSource

Ping Table\* ? wpi.dbo.Event

Continue ?

Save

At the Configure Connection Parameters page, you define the network-related information necessary for the application view to interact with the target EIS. You need to enter this information only once per application view.

7. Enter your WebLogic User Name and WebLogic Password.

**Note:** Your page may have different fields than the ones shown. The fields are determined by the adapter.

8. For any remaining fields, consult the relevant technical analyst or EIS specialist for the required information.

9. Click Continue. The Application View Administration page displays.

**Note:** To save the application view for later completion, click Save at any time.

The screenshot displays the 'Application View Administration' page for 'EastCoast.Sales.CustomerManagement'. The page includes a navigation menu on the left with options like 'Configure Connection', 'Administration', 'Add Service', 'Add Event', and 'Deploy Application View'. The main content area shows a description field, a table of connection criteria, and sections for adding events and services.

Connection criteria	
Additional Log Category:	CustomerManagement
Root Log Category:	BEA_WLS_DBMS_ADK
WebLogic Password:	security
Ping Table:	wlpi.dbo.Event
Message Bundle Base:	BEA_WLS_DBMS_ADK
Log Configuration File:	BEA_WLS_DBMS_ADK.xml
WebLogic User Name:	system
Data Source Name (JNDI):	WLAI_DataSource

Below the table, there are sections for 'Events' and 'Services', each with an 'Add' button. At the bottom, there is a 'Save App View' button and a help icon.

## Adding Services to an Application View

After you create and configure an application view, add services that support the application's functions. For information on application view services, see "Services and Events: Specialized Message Handlers."

1. While the application view is open, click Administration. The Application View Administration page displays.

## 2 Defining Application Views

**Application View Administration for EastCoast.Sales.CustomerManagement**

Adapter Home WLF Home Page WebLogic Console Glossary Logout

Configure Connection  
Administration  
Add Service  
Add Event  
Deploy Application View

*This page allows you to add events and/or services to an application view.*

**Description:** Your description here. [Edit](#)

**Connection criteria**

<b>Additional Log Category:</b>	CustomerManagement
<b>Root Log Category:</b>	BEA_WLS_DBMS_ADK
<b>WebLogic Password:</b>	security
<b>Ping Table:</b>	wlpi.dbo.Event
<b>Message Bundle Base:</b>	BEA_WLS_DBMS_ADK
<b>Log Configuration File:</b>	BEA_WLS_DBMS_ADK.xml
<b>WebLogic User Name:</b>	system
<b>Data Source Name (JNDI):</b>	WLAI_DataSource

[Reconfigure connection parameters for CustomerManagement](#)

**Events**

**Services**

?

2. Click Add Service. The Add Service page displays.

**Add Service**

Adapter Home WLF Home Page WebLogic Console Glossary Logout

Configure Connection  
Administration  
Add Service  
Add Event  
Deploy Application View

*On this page, you add services to your application view.*

**Unique Service Name:**

**Description:**

**SQL Statement:**

[Browse DBMS...](#)

Syntax Help: 1. Use fully qualified table name (i.e. catalog.schema.table); 2. to gather user input, bracket the column name and type as follows: "[ColumnName ColumnType]". Hint: browse to cut & paste ColumnName and ColumnType into your sql.

**Note:** Your page may have different fields than the ones shown. The fields are determined by the adapter.

3. In the Unique Service Name field, enter a name. The name should describe the function performed by this service. Each service name must be unique to its application view. Valid characters are a–z, A–Z, 0–9, and \_ (underscore).
4. In the Description field, enter any relevant notes. These notes are viewed by users when they use this application view service in workflows using WebLogic Process Integrator.
5. For any remaining fields, consult the relevant technical analyst or EIS specialist for the required information or format.
6. When finished, click Add.

## Adding Events to an Application View

After you create and configure an application view, add the appropriate events. For information on application view events, see “Services and Events: Specialized Message Handlers.”

1. While the application view is open, click Administration. The Application View Administration page displays.

## 2 Defining Application Views

**Add Service**

Adapter Home WLF Home Page WebLogic Console Glossary Logout

Configure Connection Administration  
Add Service  
Add Event  
Deploy Application View

On this page, you add services to your application view.

Unique Service Name:\* RetrieveAllCustomers

Description: Returns a list of all customer records, including first name, last name, and date of birth.

SQL Statement:\* select \* from wipi.dbo.Customer\_Table

Add

[Browse DBMS...](#)

Syntax Help: 1. Use fully qualified table name (i.e. catalog.schema.table); 2. to gather user input, bracket the column name and type as follows: "[ColumnName ColumnType]". Hint: browse to cut & paste ColumnName and ColumnType into your sql.

2. Click Add Event. The Add Event page displays.

**Add Event**

Adapter Home WLF Home Page WebLogic Console Glossary Logout

Configure Connection Administration  
Add Service  
Add Event  
Deploy Application View

On this page, you add events to your application view.

Unique Event Name:\* CustomerInserted

Description: This event is triggered when a new customer record is added.

Table Name:\* wipi.dbo.Customer\_Table ?

[Browse DBMS...](#)

Please Select The Type Of Event To Create:

Insert Event  
 Update Event  
 Delete Event

Add

**Note:** Your page may have different fields than the ones shown. The fields are determined by the adapter.

3. In the Unique Event Name field, enter a name. Each event name must be unique to its application view. Valid characters are a–z, A–Z, 0–9, and \_ (underscore).
4. In the Description field, enter any relevant notes. These notes are viewed by users when they use this application view event in workflows using WebLogic Process Integrator.
5. For any remaining fields, consult the relevant technical analyst or EIS specialist for the required information or format.
6. When finished, click Add. The Application View Administration page displays.

## Deploying an Application View

Deploy an application view after you have added at least one event or service to it, and you want to use or test the application view. (See “Testing an Application View’s Services” and “Testing an Application View’s Events.”) Deploy an application view whenever you want to test its services and events or use the application view in your WebLogic environment. Application view deployment places relevant metadata about its services and events is placed into a runtime metadata repository. In this way, the application view is made available to other WebLogic entities. This means other business processes can interact with the application view, and you can test the application view’s services and events.

To deploy an application view:

1. Open the application view. (See “Logging On to the Application View Management Console.”) The Summary for Application View page displays.

## 2 Defining Application Views

The screenshot displays the 'Summary for Application View EastCoast.Sales.CustomerManagement' page in the BEA WebLogic console. The page includes a navigation bar with links for 'Adapter Home', 'WLIF Home Page', 'WebLogic Console', 'Glossary', and 'Logout'. A sidebar on the left shows 'Summary' as the active menu item. The main content area contains the following information:


- Description:** This application view supports very basic operations on a customer database.
- Deployment Status:** Not Deployed
- Available Actions:** Edit, Delete, Deploy App View (with a help icon), and a checked checkbox for 'Deploy persistently?' (with a help icon).
- Summary of Existing Events and Services:**
  - Current Events for EastCoast.Sales.CustomerManagement:** CustomerInserted
    - [View Summary](#)
    - [View Event Schema](#)
  - Current Services for EastCoast.Sales.CustomerManagement:**
    - InsertCustomer
      - [View Summary](#)
      - [View Request Schema](#)
      - [View Response Schema](#)
    - RetrieveAllCustomers
      - [View Summary](#)
      - [View Request Schema](#)
      - [View Response Schema](#)

2. To automatically redeploy this application view whenever the WebLogic server is restarted, select Deploy Persistently.
3. To deploy the application view, click Deploy App View. The Deploy Application View page displays.



## Steps for Defining Application Views

### Deploy Application View EastCoast.Sales.CustomerManagement to Server



Adapter Home   WLF Home Page   WebLogic Console   Glossary   Logout

Configure Connection   Administration   Add Service   Add Event   **Deploy Application View**

*On this page you deploy your application view to the application server.*

#### Connection Pool Parameters

Use these parameters to configure the connection pool used by this application view

Minimum Pool Size\*

Maximum Pool Size\*

Target Fraction of Maximum Pool Size\*

Allow Pool to Shrink?

#### Log Configuration

Set the log verbosity level for this application view.

#### Configure Security

[Restrict Access to CustomerManagement using J2EE Security](#)

#### Required Service Parameters

Enable asynchronous service invocation?  ?

#### Required Event Parameters

Event Router URL\* ?

?  Deploy persistently? ?  ?

**Note:** On the Deploy Application View page, the actual fields you see depend on the adapter. For an explanation of all fields, consult the relevant technical analyst or EIS specialist.

## 2 Defining Application Views

---

4. In the Minimum Pool Size field, enter the minimum number of connection pools to be used by this application view. Example: 1.
5. In the Maximum Pool Size field, enter the maximum number of connection pools to be used by this application view. Example: 10.
6. In the Target Fraction of Maximum Pool Size field, enter the ideal pool size, measured from 0 to 1.0. Example: 0.7. If the Maximum Pool Size is 10 and the Target Fraction is 0.7, this means the adapter will perform load balancing to attempt to maintain the connection pool size at 70% of the maximum, which in this case means 7 connections.
7. To automatically delete unused connections, select Allow Pool to Shrink.
8. On the Log Configuration area, choose one of the following options according to your logging preferences:
  - Log errors and audit messages
  - Log warnings, errors, and audit messages
  - Log informationals, warnings, errors, and audit messages
  - Log all messages
9. If necessary, click Restrict Access using J2EE Security. The Application View Security page displays.

**Application View Security**

Adapter Home WLIIF Home Page WebLogic Console Glossary Logout

Configure Connection Administration Add Service Add Event Deploy Application View

This form allows you to grant and revoke permissions for this application view.

Choose an Action:  Grant  Revoke

Specify a User or Group:\*

Permission:  Read (Invoke Service or Register for Event)  Write (Deploy/Undeploy/Edit App View)

Principals with Read Access Granted	Principals with Read Access Revoked
<ul style="list-style-type: none"> <li>everyone</li> </ul>	no person/group specified
Principals with Write Access Granted	Principals with Write Access Revoked
<ul style="list-style-type: none"> <li>everyone</li> </ul>	no person/group specified

Apply Done

Use this page to grant or revoke a WebLogic user or group's read and write access to this application view.

10. When finished setting up permissions, click Apply to save your changes.
11. To return to the Deploy Application View page, click Done.
12. To enable other WebLogic entities, such as WebLogic Process Integrator actions, to asynchronously call the services of this application view, select Enable Asynchronous Service Invocation (applies only to application views with services).
 

An entity that calls an application view service asynchronously will continue its process without waiting for a response from the service.
13. If this application view has events, enter the URL of the adapter's event router.  
Example: `http://localhost:7001/YourEIS_EventRouter/EventRouter`
14. To automatically redeploy this application view whenever the WebLogic server is restarted, select Deploy Persistently.
15. Click Deploy App View. The Summary for Application View page displays.

**Note:** To save the application view for later completion, click Save at any time.

## Undeploying an Application View

Undeploy an application view when you want to edit its connection parameters or add services and events. For information on editing connection parameters, see “Defining an Application View.” When an application view is undeployed, no other WebLogic entities can interact with it, and you can not test its services or events.

To undeploy an application view:

1. Click Summary. The Summary for Application View page displays.

The screenshot shows the 'Summary for Application View EastCoast.Sales.CustomerManagement' page. The page title is 'Summary for Application View EastCoast.Sales.CustomerManagement'. The breadcrumb navigation includes 'Adapter Home', 'WLIF Home Page', and 'WebLogic Console'. The 'Summary' tab is selected. The page content includes a description: 'This application view supports very basic operations on a customer database.' The deployment status is 'Deployed', and there is an 'Undeploy' link. Below this, there are sections for 'Summary of Existing Events and Services', 'Current Events for EastCoast.Sales.CustomerManagement' (listing 'CustomerInserted' with links for 'View Summary', 'View Event Schema', and 'Test'), 'Current Services for EastCoast.Sales.CustomerManagement' (listing 'InsertCustomer' and 'RetrieveAllCustomers' with links for 'View Summary', 'View Request Schema', 'View Response Schema', and 'Test').

2. To undeploy the application view from the WebLogic server, click Undeploy. The Undeploy Application View page displays.



#### Undeploy Application View

*This page confirms that you want to UNDEPLOY the application view.*

Are you sure you want to undeploy  
EastCoast.Sales.CustomerManagement?

Confirm

Cancel

3. Click Confirm. The Summary for Application View page displays, indicating you may deploy the application view again.

## Testing an Application View's Services

After you create and deploy an application view that contains services, test the application view services. Testing evaluates whether or not the application view service interacts properly with the target EIS. To test application view services:

1. Define an application view (See “Defining an Application View.”), add the appropriate services, and deploy the application view (See “Deploying an Application View.”) if you have not done so already.

You can test an application view only if the application view is deployed and it contains at least one event or service.

2. On the left navigation area, click Summary. The Summary for Application View page displays.

## 2 Defining Application Views

The screenshot displays the 'Summary for Application View EastCoast.Sales.CustomerManagement' page in the BEA WebLogic console. The page includes a navigation bar with links for 'Adapter Home', 'WLIJ Home Page', 'WebLogic Console', 'Glossary', and 'Logout'. A sidebar on the left shows 'Summary' as the active section. The main content area contains the following information:

- Description:** This application view supports very basic operations on a customer database.
- Deployment Status:** Deployed
- Available Actions:** [Undeploy](#)
- Summary of Existing Events and Services:**
  - Current Events for EastCoast.Sales.CustomerManagement:**
    - CustomerInserted
      - [View Summary](#)
      - [View Event Schema](#)
      - [Test](#)
  - Current Services for EastCoast.Sales.CustomerManagement:**
    - InsertCustomer
      - [View Summary](#)
      - [View Request Schema](#)
      - [View Response Schema](#)
      - [Test](#)
    - RetrieveAllCustomers
      - [View Summary](#)
      - [View Request Schema](#)
      - [View Response Schema](#)
      - [Test](#)

3. In the Current Services area, find the service and click Test. The Test Service page displays.

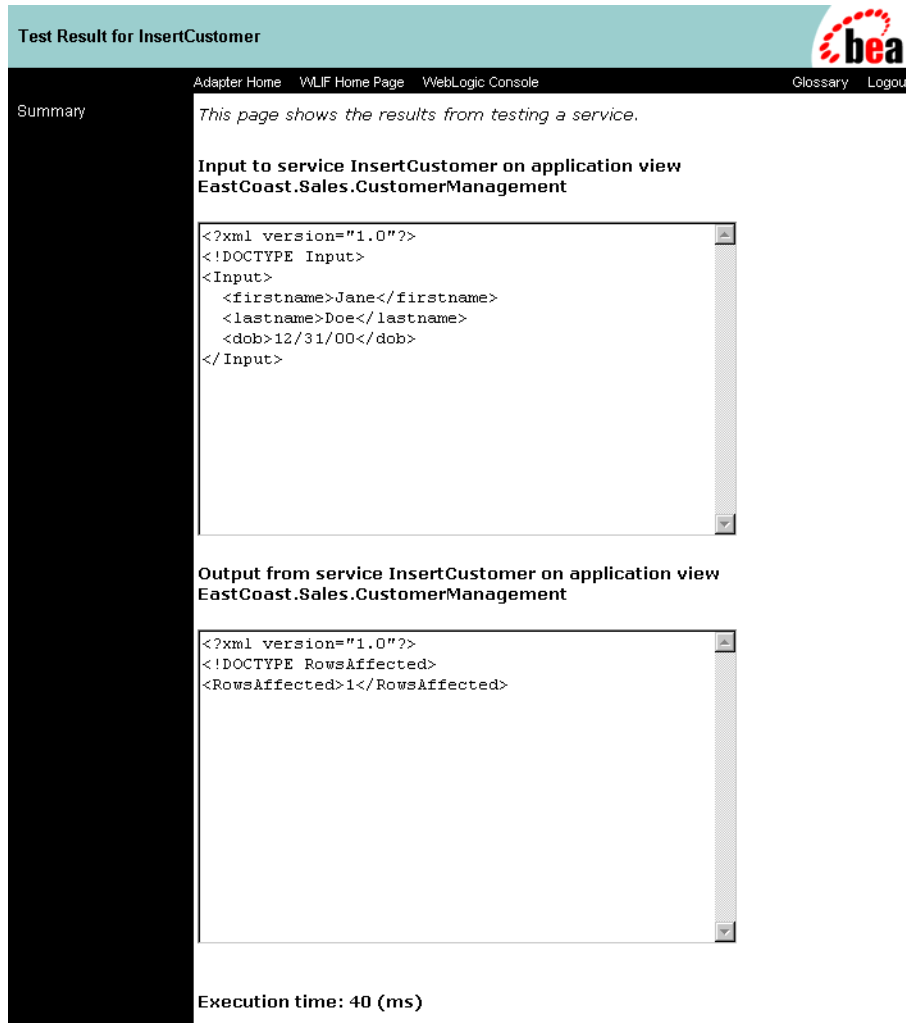
The screenshot shows the 'Test Service: InsertCustomer' page. At the top right is the BEA logo. Below the title bar, there are navigation links: 'Adapter Home', 'WLIF Home Page', 'WebLogic Console', 'Glossary', and 'Logout'. The main content area is divided into a 'Summary' sidebar and a main panel. The main panel contains the instruction: 'Please fill in any inputs to the service query and click Test'. Below this is the 'Application View: InsertCustomer EastCoast.Sales.CustomerManagement' section, which displays the SQL query: 'insert into wpl.dbo.Customer\_Table (firstname,lastname,dob) values([firstname varchar],[lastname varchar],[dob varchar])'. Underneath the query is an 'Input' section with three text fields: 'firstname' (containing 'Jane'), 'lastname' (containing 'Doe'), and 'dob' (containing '12/31/00'). A 'Test' button is located at the bottom of the input section.

4. If necessary, enter the service input data in the Input fields. If the application view service processes this data correctly, the test is successful.

**Note:** Your Test Service page may have different fields than the ones shown. The fields are determined by the application view service. For an explanation of the fields, consult the relevant technical analyst or EIS specialist.

5. Click Test after entering the service input data. The Test Result page displays. This page displays the input and output documents.

## 2 Defining Application Views



The screenshot displays the 'Test Result for InsertCustomer' page in the WebLogic Console. The page includes a navigation bar with links for 'Adapter Home', 'WLIF Home Page', 'WebLogic Console', 'Glossary', and 'Logout'. The BEA logo is visible in the top right corner. The main content area is divided into a 'Summary' section on the left and a main test result area on the right. The summary section contains the text 'This page shows the results from testing a service.' The main test result area is titled 'Input to service InsertCustomer on application view EastCoast.Sales.CustomerManagement' and shows the following XML input:

```
<?xml version="1.0"?>
<!DOCTYPE Input>
<Input>
  <firstname>Jane</firstname>
  <lastname>Doe</lastname>
  <dob>12/31/00</dob>
</Input>
```

Below the input, the output is titled 'Output from service InsertCustomer on application view EastCoast.Sales.CustomerManagement' and shows the following XML output:

```
<?xml version="1.0"?>
<!DOCTYPE RowsAffected>
<RowsAffected>1</RowsAffected>
```

At the bottom of the test result area, the execution time is displayed as 'Execution time: 40 (ms)'.

6. Repeat the test procedure for each service you want to test.
7. When finished testing the application view's services, you may keep the application view deployed or undeploy it (See "Undeploying an Application View.") to edit the application view.



## Testing an Application View's Events

After creating and deploying an application view that contains events, test the application view events. Testing evaluates whether or not the application view responds correctly to the EIS application. To test application view events:

1. Define an application view (See “Defining an Application View.”), add the appropriate events, and deploy the application view (See “Deploying an Application View.”) if you have not done so already.

You can test an application view only if it is deployed and contains at least one event or service.

2. Click Summary. The Summary for Application View page displays.

**Summary for Application View EastCoast.Sales.CustomerManagement**

Adapter Home WLF Home Page WebLogic Console Glossary Logout

**Summary**

*This page shows the events and services defined for the **EastCoast.Sales.CustomerManagement** application view.*

**Description:** This application view supports very basic operations on a customer database.

**Deployment Status:** Deployed

**Available Actions:** [Undeploy](#)

---

**Summary of Existing Events and Services:**

**Current Events for EastCoast.Sales.CustomerManagement**

CustomerInserted

- [View Summary](#)
- [View Event Schema](#)
- [Test](#)

**Current Services for EastCoast.Sales.CustomerManagement**

InsertCustomer

- [View Summary](#)
- [View Request Schema](#)
- [View Response Schema](#)
- [Test](#)

RetrieveAllCustomers

- [View Summary](#)
- [View Request Schema](#)
- [View Response Schema](#)
- [Test](#)

## 2 Defining Application Views

3. In the Current Events area, find your event and click Test. The Test Event page displays.

**Test Event: CustomerInserted**

Adapter Home WLIF Home Page WebLogic Console Glossary Logout

**Summary**

*This page allows you to test an event. This page allows you to create the event by invoking a service that will cause the event, or manually using EIS-specific tools.*

If you want to use a service invocation to create the event, select the 'Service' option below, and select the service to invoke. Otherwise, you can create the event manually using any tools your EIS provides (for example an interactive SQL tool for the DBMS adapter used to insert a new row and create the event).

How do you want to create the event?

Service

Manual

How long should we wait to receive the event?

Time (in milliseconds):

**Note:** Your Test Event page may have different fields than the ones shown. The fields are determined by the application view service. For an explanation of the fields, consult the relevant technical analyst or EIS specialist.

4. Choose the method to use to generate the test event:
  - Service (see “If you choose Service”): Choose Service when you want to use one of the application view’s own services to generate a “canned” event.
  - Manual (see “If you choose Manual”): Choose Manual when you want to generate the event by logging on to an EIS application and perform the appropriate event-generating function.

If the application view event correctly responds before the specified time elapses, the test is successful.

### If you choose Service

- a. On the Service menu, choose a service that will trigger the event you are testing. Example: If you are testing the “NewCustomer” event, choose a service that will invoke it, such as “Insert Customer.”

- b. In the Time field, enter a reasonable time to wait, in milliseconds. If this time elapses before the event succeeds, the test will time out and display a failure message.
- c. Click “Test.” The triggering service is executed.  
If the service requires input data, an input page displays.

**Test Service: InsertCustomer**

Adapter Home   WLF Home Page   WebLogic Console   Glossary   Logout

Summary   *Please fill in any inputs to the service query and click Test*

**Test Event: InsertCustomer EastCoast.Sales.CustomerManagement**

```
insert into wlpi.dbo.Customer_Table (firstname,lastname,dob) values([firstname
varchar],[lastname varchar],[dob varchar])
```

**Input**

**firstname**  text

**lastname**  text


**dob**  text

- d. If necessary, enter the service input data in the fields, then click Test.

The service executes. If the test succeeds, the Test Result page displays. A successful test result displays the event document, the service input document, and the service output document.

## 2 Defining Application Views

**Test Result for CustomerInserted**

Adapter Home   WLIJ Home Page   WebLogic Console      Glossary   Logout

Summary   *This page shows the results from testing a event.*

**Generated event of type CustomerInserted on application view EastCoast.Sales.CustomerManagement**

```
<?xml version="1.0"?>
<!DOCTYPE Customer_Table.insert>
<Customer_Table.insert>
  <Address1/>
  <Address2/>
  <Address3/>
  <City/>
  <Country/>
  <DOB>Dec 31 1999 12:00AM</DOB>
  <Email/>
  <Fax/>
  <FirstName>John</FirstName>
  <LastName>Doe</LastName>
  <MiddleName/>
  <Phone/>

```

**Input to service InsertCustomer on application view EastCoast.Sales.CustomerManagement**

```
<?xml version="1.0"?>
<!DOCTYPE Input>
<Input>
  <firstname>John</firstname>
  <lastname>Doe</lastname>
  <dob>12/31/99</dob>
</Input>

```

**Output from service InsertCustomer on application view EastCoast.Sales.CustomerManagement**

```
<?xml version="1.0"?>
<!DOCTYPE RowsAffected>
<RowsAffected>1</RowsAffected>

```

Execution time: 3034 (ms)

If the test fails, the Test Result page displays only a “Timed Out” message.

**Test Result for CustomerInserted**

Adapter Home WLF Home Page WebLogic Console Glossary Logout

Summary

*This page shows the results from testing a event.*

**Generated event of type CustomerInserted on application view EastCoast.Sales.CustomerManagement**

Timed Out

Execution time: 6028 (ms)

- e. If the test failed, edit the event definition, or contact the system administrator or application manager.
- f. If the test succeeded, repeat the test procedure for each remaining event you want to test.
- g. When finished, save the application view.

### If you choose Manual

- a. In the Time field, enter a reasonable time to wait, in milliseconds. (One minute = 60,000 ms.) If this time elapses before the event succeeds, the test will time out and display a failure message.
- b. Open the application you will use to trigger the event, if the application is not already open.
- c. Click “Test.” The test waits for an event to trigger it.
- d. Using the triggering application, perform an action that will execute the service that will test the application view event.

If the test succeeds, the Test Result page displays. A successful test result displays the event document from the application, the service input document, and the service output document.

## 2 Defining Application Views

Test Result for CustomerInserted 

Adapter Home WLIJ Home Page WebLogic Console [Glossary](#) [Logout](#)

Summary *This page shows the results from testing a event.*

**Generated event of type CustomerInserted on application view EastCoast.Sales.CustomerManagement**

```
<?xml version="1.0"?>
<!DOCTYPE Customer_Table.insert>
<Customer_Table.insert>
  <Address1/>
  <Address2/>
  <Address3/>
  <City/>
  <Country/>
  <DOB>Dec 31 1999 12:00AM</DOB>
  <Email/>
  <Fax/>
  <FirstName>John</FirstName>
  <LastName>Doe</LastName>
  <MiddleName/>
  <Phone/>
```

**Input to service InsertCustomer on application view EastCoast.Sales.CustomerManagement**

```
<?xml version="1.0"?>
<!DOCTYPE Input>
<Input>
  <firstname>John</firstname>
  <lastname>Doe</lastname>
  <dob>12/31/99</dob>
</Input>
```

**Output from service InsertCustomer on application view EastCoast.Sales.CustomerManagement**

```
<?xml version="1.0"?>
<!DOCTYPE RowsAffected>
<RowsAffected>1</RowsAffected>
```

Execution time: 3034 (ms)

If the test fails or takes too long, the Test Result page displays containing a “Timed Out” message.



**Test Result for CustomerInserted**

Adapter Home WLF Home Page WebLogic Console Glossary Logout

Summary

*This page shows the results from testing a event.*

**Generated event of type CustomerInserted on application view EastCoast.Sales.CustomerManagement**

Timed Out

Execution time: 6028 (ms)

- e. If the test failed, edit the event definition, or contact the system administrator or application manager.
- f. If the test succeeded, repeat the test procedure for each remaining event you want to test.
- g. When finished, save the application view.

## 2 Defining Application Views

The screenshot shows the 'Summary for Application View EastCoast.Sales.CustomerManagement' page. The page title is 'Summary for Application View EastCoast.Sales.CustomerManagement' and the BEA logo is in the top right corner. The page content includes a navigation bar with 'Adapter Home', 'WLIJ Home Page', and 'WebLogic Console', and 'Glossary' and 'Logout' links. A left sidebar contains a 'Summary' link. The main content area starts with a paragraph: 'This page shows the events and services defined for the EastCoast.Sales.CustomerManagement application view.' Below this are sections for 'Description:', 'Deployment Status:', and 'Available Actions:'. The 'Description:' section states: 'This application view supports very basic operations on a customer database.' The 'Deployment Status:' is 'Deployed' and 'Available Actions:' includes a link to 'Undeploy'. A section titled 'Summary of Existing Events and Services:' follows. Under 'Current Events for EastCoast.Sales.CustomerManagement', there is a list for 'CustomerInserted' with links for 'View Summary', 'View Event Schema', and 'Test'. Under 'Current Services for EastCoast.Sales.CustomerManagement', there are two lists: 'InsertCustomer' with links for 'View Summary', 'View Request Schema', 'View Response Schema', and 'Test'; and 'RetrieveAllCustomers' with links for 'View Summary', 'View Request Schema', 'View Response Schema', and 'Test'.

## Editing an Application View

When you define an application view, you must configure its connection parameters. After you add and test services and events, you may want to reconfigure the connection parameters or remove services and events. To edit an existing application view:

1. Open the application view.
2. Click Summary. The Summary for Application View page displays.



The screenshot displays the 'Summary for Application View EastCoast.Sales.CustomerManagement' page in the BEA WebLogic console. The page includes a navigation bar with links for 'Adapter Home', 'WLF Home Page', 'WebLogic Console', 'Glossary', and 'Logout'. A sidebar on the left is labeled 'Summary'. The main content area contains the following information:

- Description:** This application view supports very basic operations on a customer database.
- Deployment Status:** Not Deployed
- Available Actions:** Edit, Delete, Deploy App View (with a help icon), and a checked checkbox for 'Deploy persistently?' (with a help icon).
- Summary of Existing Events and Services:**
  - Current Events for EastCoast.Sales.CustomerManagement:** CustomerInserted
    - View Summary
    - View Event Schema
  - Current Services for EastCoast.Sales.CustomerManagement:**
    - InsertCustomer
      - View Summary
      - View Request Schema
      - View Response Schema
    - RetrieveAllCustomers
      - View Summary
      - View Request Schema
      - View Response Schema

3. In the Available Actions area, click Edit. The Application View Administration page displays.

## 2 Defining Application Views

The screenshot displays the 'Application View Administration' interface for 'EastCoast.Sales.CustomerManagement'. The left sidebar contains navigation options: 'Configure Connection', 'Administration' (selected), 'Add Service', 'Add Event', and 'Deploy Application View'. The main content area includes a breadcrumb trail (Adapter Home, WLIF Home Page, WebLogic Console), a 'Glossary' and 'Logout' link, and a descriptive paragraph: 'This page allows you to add events and/or services to an application view.' Below this is a 'Description' section stating: 'This application view supports very basic operations on a customer database. [\\_Edit](#)'.

The 'Connection criteria' section is highlighted and contains the following details:

Message Bundle Base:	BEA_WLS_DBMS_ADK
Data Source Name (JNDI):	WLAI_DataSource
Additional Log Category:	CustomerManagement
WebLogic User Name:	system
WebLogic Password:	security
Root Log Category:	BEA_WLS_DBMS_ADK
Ping Table:	wlpi.dbo.Event
Log Level:	WARN
Log Configuration File:	BEA_WLS_DBMS_ADK.xml

A link to 'Reconfigure connection parameters for CustomerManagement' is provided below the criteria table.

The 'Events' section shows one event, 'CustomerInserted', with links for 'Remove Event', 'View Summary', and 'View Event Schemas', and an 'Add' button.

The 'Services' section shows two services, 'InsertCustomer' and 'RetrieveAllCustomers', each with links for 'Remove Service', 'View Summary', 'View Request Schema', and 'View Response Schema', and an 'Add' button.

At the bottom, there are buttons for 'Continue To Deploy' and 'Save App View' with a help icon.

4. To reconfigure the application view's connection parameters, click Configure Connection (See "Defining an Application View.")
5. To add services and events, click Add Service (See "Adding Services to an Application View.") or Add Event (See "Adding Events to an Application View.").

# 3 Using Application Views in WebLogic Process Integrator

This section contains information on the following subjects:

- Before You Begin
- Introduction to Using Application Views in WebLogic Process Integrator
- Using Application Views in WebLogic Process Integrator
  - Scenario 1: Setting Up a Task Node to Call an Application View Service
  - Scenario 2: Setting Up an Event Node to Wait for a Response from an Asynchronous Application View Service
  - Scenario 3: Creating a Workflow that is Started by an Application View Event
  - Scenario 4: Setting Up an Event Node to Wait for an Application View Event

## Before You Begin

The following prerequisites must have been met before you can invoke an application view service or receive an application view event in WebLogic Process Integrator:

- You have created an application view and defined services and events for the application view.
- The application view and its adapter are functional and saved. If you plan on calling application view services and events from a running workflow, the application view must be deployed, as well.
- WebLogic Process Integrator is running.
- WebLogic Application Integration is running.
- The Application Integration Plug-in (AI Plug-in) has been loaded.
- You have received information about the required business logic for the workflows you are defining. This information usually comes from the business analyst or someone similar.
- A workflow template definition is open.

# Introduction to Using Application Views in WebLogic Process Integrator

After you create all the required application view services and events for your enterprise, use the application views to execute your business processes. The simplest way to do this is by using WebLogic Process Integrator to design workflows that use the application view services and events.

WebLogic Process Integrator provides a GUI-based environment for designing business process workflows. These workflows can include application view services and events defined using Application Integration. For complete information on Process Integrator, see “BEA WebLogic Process Integrator.”

## Using Application Views in WebLogic Process Integrator

There are four ways to use application view services and events in WebLogic Process Integrator:

- Scenario 1: Setting Up a Task Node to Call an Application View Service
- Scenario 2: Setting Up an Event Node to Wait for a Response from an Asynchronous Application View Service
- Scenario 3: Creating a Workflow that is Started by an Application View Event
- Scenario 4: Setting Up an Event Node to Wait for an Application View Event

Use these scenarios in combination with each other to create your own workflows. This document does not fully explain how to use WebLogic Process Integrator. For complete information on WebLogic Process Integrator, see the *WebLogic Process Integrator User Guide* or see <http://edocs.bea.com>.

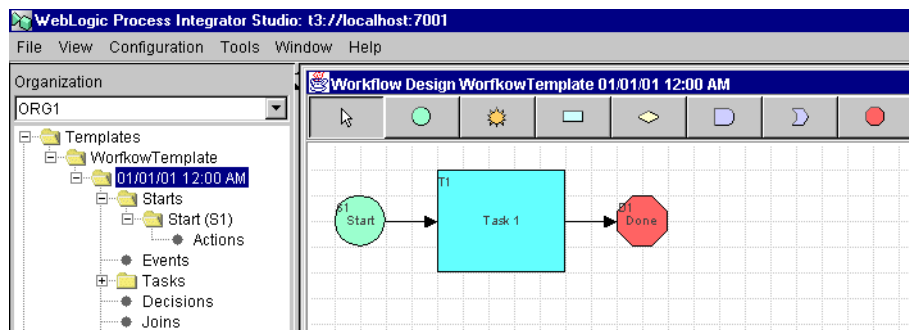
## Scenario 1: Setting Up a Task Node to Call an Application View Service

In your organization, there may be situations in which you want to call an application view service from within a workflow. To do this, add a task node to the workflow, then add an appropriate Application View Service action to the task node. When the workflow is saved and activated, the application view service will be called whenever the task node executes.

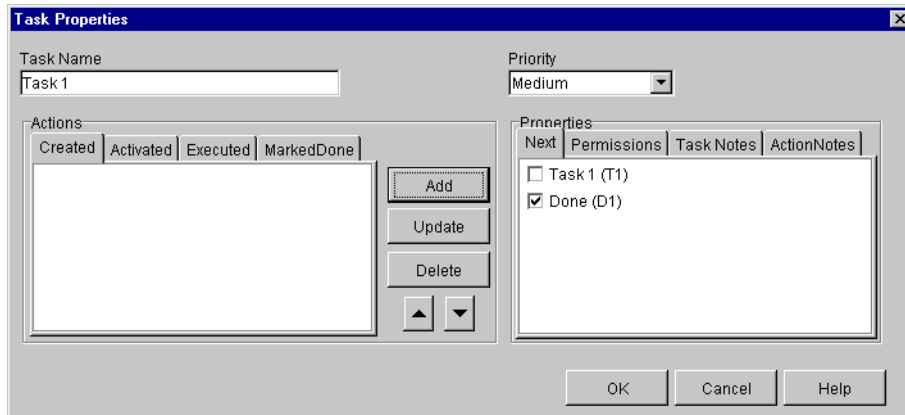
### Steps for Setting up a Task Node to Call an Application View Service

Follow these steps to create a task node that calls an application view service:

1. Within WebLogic Process Integrator Studio, open a template definition. The Workflow Design window displays.



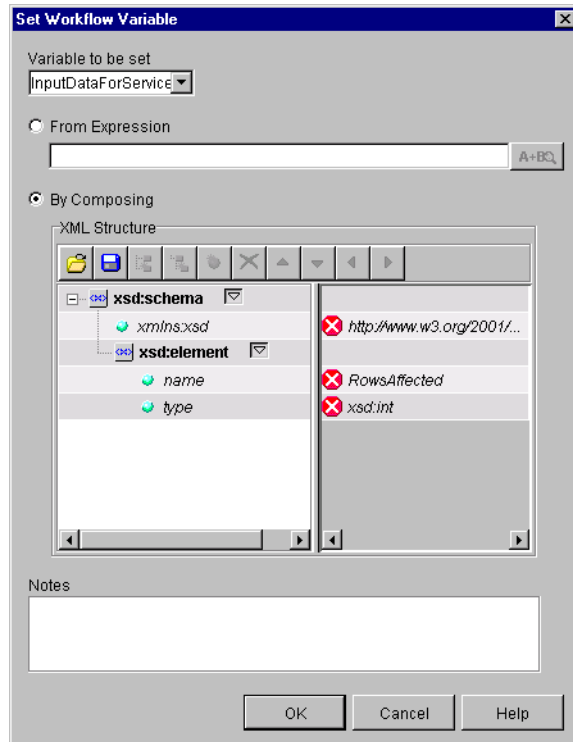
2. Create a task node if one does not already exist.
3. Double-click the task node that will call the application view service. The Task Properties dialog box displays.



4. In the Actions area, select the tab from which you want the service to be called. Your tab choice depends on your business processes.
5. Click Add. The Add Action dialog box displays.



6. From the Action category tree, select Workflow Actions, Set Workflow Variable, and click OK. The Set Workflow Variable dialog box displays.

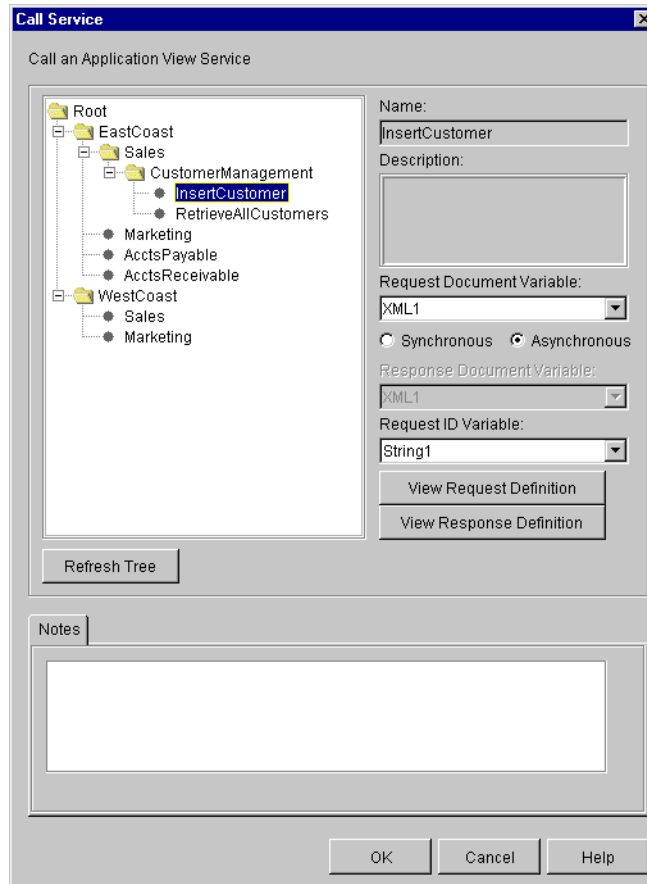


7. In the Variable To Be Set list, select an XML variable to contain the input data for the application view service.
8. Click OK.
9. On the Task Properties dialog box, click Add. The Add Action dialog box displays.





10. From the action tree, select AI Actions, Call Application View Service and click OK. The Call Service dialog box displays.



11. In the application view tree, navigate to and select the service you want to call.

The application view tree organizes application view services by folder (example: EastCoast.Sales) and application view (example: CustomerManagement). All application view services are at the lowest level of the hierarchy.

**Note:** To check for newly saved application views and events at any time, click Refresh Tree.

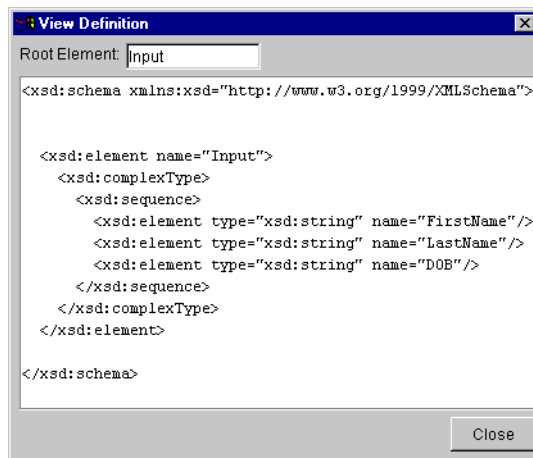
12. In the Request Document Variable list, select an XML variable.

## Using Application Views in WebLogic Process Integrator

---

Because this XML variable serves as the input document of the application view service, make sure the variable is properly set before the service is called.

**Note:** If you need to examine the XML schema of the input document, click View Request Definition. The View Definition dialog box displays. Click Close when finished.

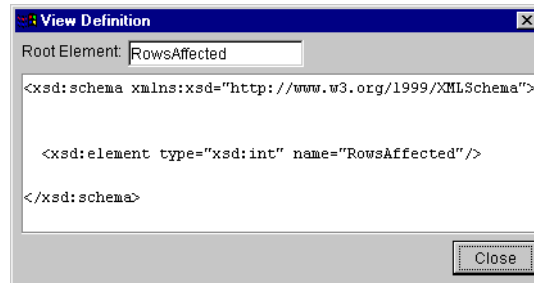


13. To call the application view synchronously, select Synchronous, or select Asynchronous to call the application view asynchronously.

**Note:** A node that synchronously calls a service will wait for the service to return a response document before the workflow can continue. If the node asynchronously calls a service, the workflow will continue.

14. For synchronous services that require a response, select an already-defined XML variable in the Response Document Variable list. When WebLogic Process Integrator calls the application view service, a response is returned. When WebLogic Process Integrator receives the response from the application view service, the Response Document Variable stores the response. If you do not care about the response, skip this step.

**Note:** If you need to examine the XML schema of the response document, click View Response Definition. The View Definition dialog box displays. Click Close when finished.



15. For asynchronous services that require a response, select an already-defined string variable in the Request ID Variable list.

When the node calls the application view service, the node does not wait for a response and allows the workflow to continue processing. The workflow uses the Request ID Variable to allow asynchronous event nodes to receive the service response. If you do not care about the asynchronous service response, skip this step.

**Note:** When you set up a task node to call an asynchronous application view service, the result will be returned to WebLogic Process Integrator. The workflow identifies this response using the Request ID Variable you selected. In a typical workflow, you will want to set up a corresponding event node that waits for this response. For more information on creating such an event node, see “Scenario 2: Setting Up an Event Node to Wait for a Response from an Asynchronous Application View Service.”

16. Click OK to save the action.
17. On the Task Properties dialog box, click OK to save the node.

## Scenario 2: Setting Up an Event Node to Wait for a Response from an Asynchronous Application View Service

This section explains how to receive an asynchronous application view service response and handle any errors it may contain.

## Receiving an Asynchronous Application View Service Response

In a workflow, whenever an action calls an application view service asynchronously (see “Scenario 1: Setting Up a Task Node to Call an Application View Service”), the application view service will return a response. Normally, if you care about the response, you will want to set up a corresponding asynchronous event node to wait for the response. This section explains a highly simplified scenario in which an event node receives an application view service response without checking for errors.

## Handling Errors in an Asynchronous Application View Service Response

Although this scenario does not handle errors returned in the application view service response, you will normally want to handle errors in your own workflows. To handle asynchronous service response errors in your workflows, use the new features included in the AI Plug-in.

The AI Plug-in includes a new variable type, `AsyncServiceResponse`, and three new functions:

- `AIHasError()`
- `AIGetErrorMsg()`
- `AIGetResponseDocument()`

For complete documentation of these functions, see “Explanation of Functions Provided by the AI Plug-in.”

## Steps for Setting Up an Event Node to Wait for a Response from an Asynchronous Application View Service

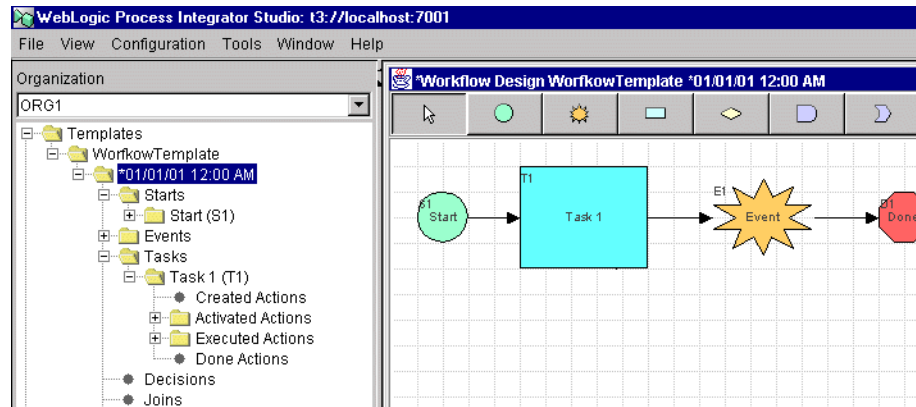
To set up an asynchronous event node to wait for a response from an asynchronous application view service, create an event node, then set up the event node to wait for an event of type “AI AsyncResponse.” When you set up an event node to wait for an asynchronous response from an application view service, the event node uses a designated Request ID Variable to receive the response. As long as the service-calling task node and the response-receiving event node use the same Request ID Variable, the asynchronous event node will correctly receive the service response.

## Chapter

---

Follow these steps to set up an event node to wait for a response from an asynchronous application view service:

1. Within WebLogic Process Integrator Studio, open a workflow template definition. The Workflow Design window displays.



2. Create an event node if one does not already exist. This event node will wait for an asynchronous response from a designated application view service.
3. Double-click the event node. The Event Properties dialog box displays.

The screenshot shows the 'Event Properties' dialog box. The 'Description' field contains 'Event' and the 'Type' dropdown is set to 'AI Async Response'. The 'Request ID Variable' dropdown is set to 'String1' and the 'Asynchronous Service Response Variable' dropdown is set to 'WLAIASR1'. The 'Variables' tab is selected, showing an empty table with columns 'Variable' and 'Expression'. Buttons for 'Add', 'Update', and 'Delete' are visible next to the table. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

4. In the Description field, enter a name (optional).
5. In the Type list, select AI Async Response.
6. In the Request ID Variable list, select an already-defined string variable. WebLogic Process Integrator will listen for an asynchronous response with an ID matching this variable.

**Note:** The purpose of this event node is to wait for a response to a Call Application View Service action that was called asynchronously earlier in the workflow. The Call Application View Service action sets the Request ID Variable. To make the action and this event node work together, they must both use the same Request ID Variable. For more information on setting up the Call Application View Service action, see “Scenario 1: Setting Up a Task Node to Call an Application View Service.”

7. In the Asynchronous Service Response Variable list, select a variable to store the response data itself. It must be of type `AsyncServiceResponse`. If you do not care about the response data, skip this step.
8. Click OK to save the event node.

## Explanation of Functions Provided by the AI Plug-in

When using the Application Integration Plug-in, use the new functions `AIHasError()`, `AIGetErrorMsg()`, and `AIGetResponseDocument()` to interrogate AI Asynch Response. If the Application Integration Plug-in is installed in WebLogic Process Integrator, then you have access to these new functions. Using these functions, you can set up decision nodes to handle success and failure conditions.

### AIHasError()

Use `AIHasError()` to determine the status of an asynchronous service response.

Operands:

`AsyncServiceResponse` variable

Preconditions:

You have created a variable of type `AsyncServiceResponse`. You have called an asynchronous application view service. The application view service has returned a response, which is stored in your `AsyncServiceResponse` variable.

Returns:

Boolean

Output explanation:

False: The asynchronous application view service call was successful.

True: The asynchronous application view service call failed.

### AIGetErrorMsg()

Use `AIGetErrorMsg()` to retrieve the error message string returned by an asynchronous application view service.



**Operands:**

`AsyncServiceResponse` variable

**Preconditions:**

You have created a variable of type `AsyncServiceResponse`. You have called an asynchronous application view service. The application view service has returned a response, which is stored by your `AsyncServiceResponse` variable.

**Returns:**

String

**Output explanation:**

Error string: Returns an error string explaining why the asynchronous application view response failed.

Empty string: There was no error.

### `AIGetResponseDocument()`

Use `AIGetResponseDocument()` to retrieve the actual XML response document returned by an asynchronous application view service.

**Operands:**

`AsyncServiceResponse` variable

**Preconditions:**

You have created a variable of type `AsyncServiceResponse`. You have called an asynchronous application view service. The application view service has returned a response, which is stored by your `AsyncServiceResponse` variable.

**Returns:**

XML

**Output explanation:**

XML document: Returns an XML document representing the asynchronous service response.

Null: No response document was returned, because an error occurred.

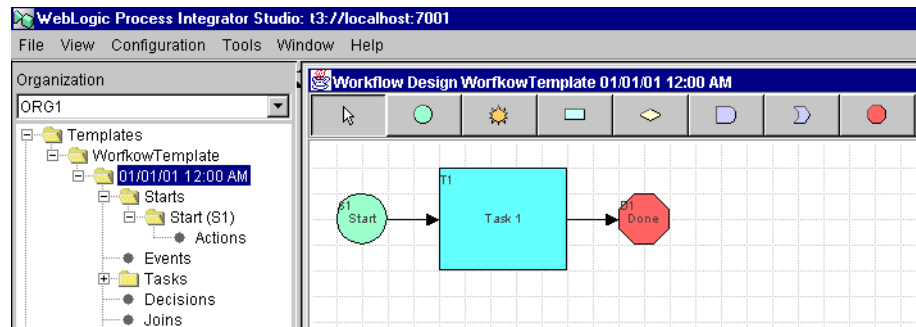
## Scenario 3: Creating a Workflow that is Started by an Application View Event

You may want to create a workflow that starts whenever a designated application view event occurs. To set up a workflow to be started by an application view event, edit the workflow's start node so it responds to an event of type AI Start, then select the appropriate application view event. If necessary, you can set up conditions on which to filter the event. After you save and activate the workflow, the start node will execute each time the application view event occurs.

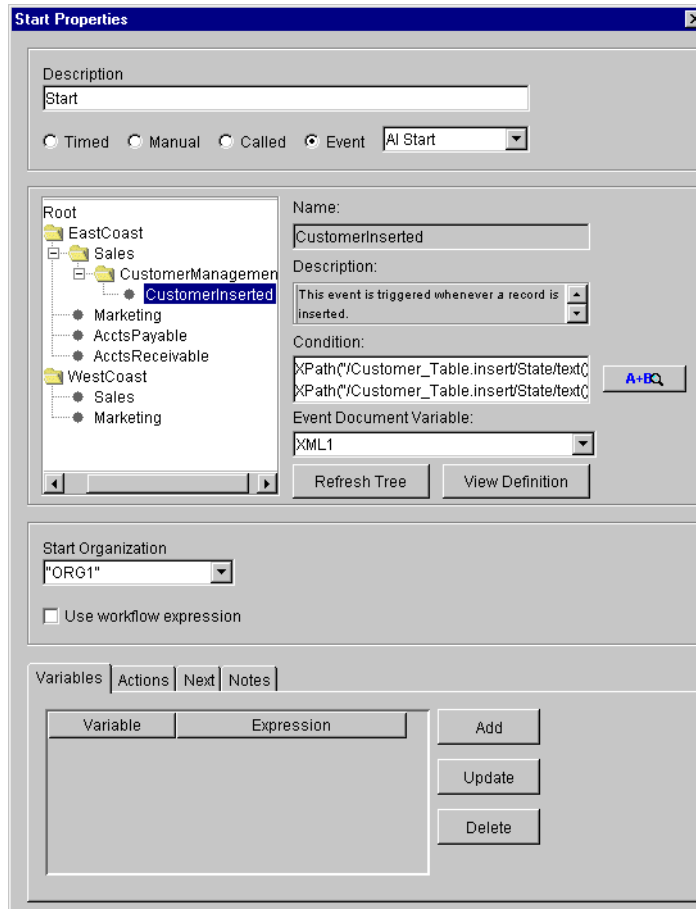
### Steps for Creating a Workflow that is Started by an Application View Event

Follow these steps to set up a workflow with a start node that is triggered by an application view event.

1. Within WebLogic Process Integrator Studio, open a template definition. The Workflow Design window displays.



2. Create a start node if one does not already exist. This start node will respond to an application view event that you specify.
3. Double-click the start node. The Start Properties dialog box displays.



4. In the Description field, enter a name (optional).
5. Click Event.
6. In the Event list, select AI Start.
7. In the application view tree, navigate to and select the application view event.

The application view tree organizes application view events by folder (example: EastCoast.Sales) and application view (example: CustomerManagement). All application view events are at the lowest level of the hierarchy.

8. If necessary, filter the event by entering a condition in the Condition field, or click the A + B button to display the Expression Builder dialog box.

For information on setting up conditions and XPath expressions, see the *WebLogic Process Integrator User Guide*.

9. In the Event Document Variable list, select an XML variable. When the start node receives data from the application view event, this variable stores the data. If you do not care about the event data, skip this step.

**Note:** If you need to examine the XML schema of the event document, click View Definition. The View Definition dialog box displays. Click Close when finished.



10. Click OK. The start node is saved.

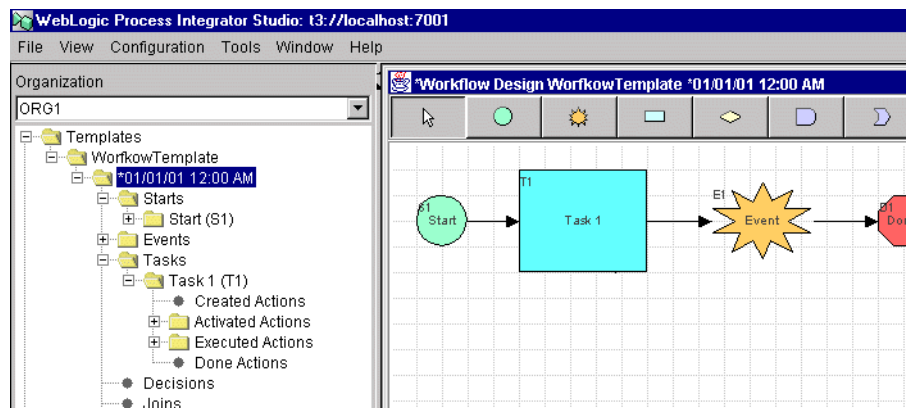
## Scenario 4: Setting Up an Event Node to Wait for an Application View Event

In a workflow, you may want to create an event node that is triggered by an application view event. To set up an event node to respond to an application view event, edit the event node so it responds to an event of type AI Event, then select the appropriate application view event. If necessary, you can set up conditions on which to filter the application view event. After you save and activate the workflow, the workflow will progress to this event node, wait for a specified application view event, and continue processing.

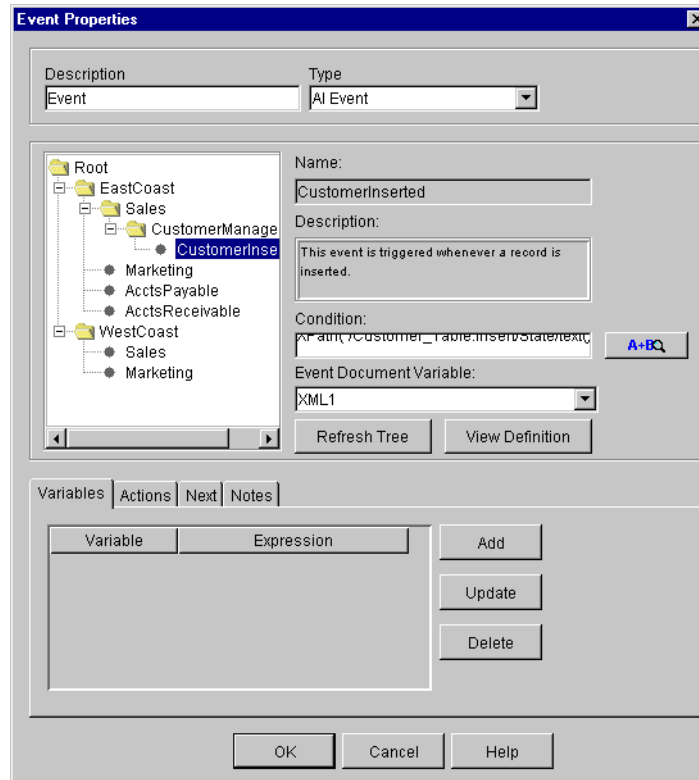
### Steps for Setting Up a Node to Wait for an Application View Event

Follow these steps to set up an event node to be triggered by an application view event.

1. Within WebLogic Process Integrator Studio, open a template definition. The Workflow Design window displays.



2. Create an event node if one does not already exist. This event node will be triggered by a designated application view event.
3. Double-click the event node. The Event Properties dialog box displays.



4. In the Description field, enter a name (optional).
5. In the Type list, select AI Event.
6. In the application view tree, navigate to and select an application view event.
 

The application view tree organizes application view events by folder (example: EastCoast.Sales) and application view (example: CustomerManagement). All application view events are at the lowest level of the hierarchy.

**Note:** To check for newly saved application views and events at any time, click Refresh Tree.
7. If necessary, filter the event by entering a condition in the Condition field, or click the A + B button to display the Expression Builder dialog box.

For information on setting up conditions and XPath expressions, see the *WebLogic Process Integrator User Guide*.

8. On the Event Properties dialog box, select an XML variable in the Event Document Variable list. When the event node receives data from the application view event, this variable stores the data. If you do not care about the event data, skip this step.

**Note:** If you need to examine the XML schema of the event document, click View Definition. The View Definition dialog box displays. Click Close when finished.



9. On the Event Properties dialog box, click OK.

Chapter

---



# 4 Using Application Views by Writing Custom Code

This section contains information on the following subjects:

- Before You Begin
- Introduction to Using Application Views by Writing Custom Code
- Steps for Using Application Views by Writing Custom Code
  - About this Example
  - Prerequisites for this Example
  - Writing the Java Class
  - Example Code for SyncCustomerInformation

# Before You Begin

The following prerequisites must have been met before you write custom Java code to implement a business process:

- You have created an application view using the Integration Framework and have defined one or more events or services within the application view.
- You have information about the required business logic for the business process workflow you are defining. This information usually comes from a business analyst. You have all the information necessary to connect to the WebLogic server, including the host server name and port number, WebLogic user ID and password.

# Introduction to Using Application Views by Writing Custom Code

Although the primary way to use application views in business processes is to use WebLogic Process Integrator (see “Using Application Views in WebLogic Process Integrator”), an alternate way is to write custom Java code to represent the business process. If you are a developer who uses the custom coding method, this section uses a simple example to demonstrate how to custom code your enterprise’s business process.

For a thorough comparison of the two ways to use application views, see “Deciding Which of the Two Methods to Use.”

## Steps for Using Application Views by Writing Custom Code

This section uses a concrete example class called `SyncCustomerInformation` to explain how to write custom code. In general, you must do the following two steps to create custom code that uses an application view in a business process:

- Make sure a Java class exists to represent the application that implements the business process.
- Within this Java class, supply the code to implement the business logic.

### About this Example

In the simple example used throughout this section, the following business logic is implemented:

An enterprise has a customer relationship management (CRM) system and an order processing (OP) system. You want a business process that coordinates the synchronization of customer information between these two systems. That means that whenever a customer is created on the CRM system, it should trigger the creation of a corresponding customer record on the OP system. The attached Java class `SyncCustomerInformation` implements this business logic.

This is not a sophisticated example. It does not cover everything you can do using custom code. It only demonstrates the basic steps you will take when you implement your own organization's business processes.

Your role is to use this example code as a template for custom coding your own business processes.

### Prerequisites for this Example

This example assumes the following prerequisites are already complete:

## 4 Using Application Views by Writing Custom Code

---

- Application views for the source CRM system and the target OP system are already defined and working. For details on defining application views, see “Defining Application Views.”
- Both of the application views exist in the “East Coast” folder. The source application view is named “East Coast.Customer Mgmt” and the target application view is named “East Coast.Order Processing.”

**Note:** Your organization will have its own folders and application views.

- You are familiar with the WebLogic Application Integration API or are working closely with a Java programmer who is.
- You have collected all the information necessary to connect to the WebLogic Application Integration server that hosts the application views.

**Note:** For your organization, get this information from the system administrator.

## Writing the Java Class

When writing custom code, there must exist a Java class to represent each application required for the business process. Create the necessary Java classes if they do not exist already. This example calls for one application class called `SyncCustomerInformation`. Of course, your own code will use different variable names. To create the `SyncCustomerInformation` Java class:

1. See “Example Code for `SyncCustomerInformation`” for the complete source code for the Java application class.

**Note:** For your own projects, use the `SyncCustomerInformation` code as a template or guide. The `SyncCustomerInformation` example code is thoroughly commented.

2. Make sure the code does the following things:
  - a. Create code to listen for East Coast.New Customer.
  - b. Obtain a reference to the `NamespaceManager` (variable name `m_namespaceMgr`) and `ApplicationViewManager` (variable name `m_appViewMgr`) within WebLogic. Accomplish this using a JNDI lookup from the WebLogic server.

## Steps for Using Application Views by Writing Custom Code

---

- c. Using the `NamespaceManager`, obtain a reference to the “root” namespace by calling `nm.getRootNamespace()`. This reference is stored in a variable called `root`.
- d. Using the `root` variable, obtain a reference to the East Coast namespace by calling `root.getNamespaceObject("East Coast")`. This reference is stored into a variable called `eastCoast`.
- e. Using the `eastCoast` variable, obtain a temporary reference to the Customer Management `Application View` and store it into a variable called `custMgmtHandle`.
- f. This `custMgmtHandle` temporary reference will be used to obtain an actual reference to an `Application View` instance for Customer Management. Do this by calling the `Application View Manager` as `avm.getApplicationViewInstance(custMgmtHandle.getQualifiedName())`. Store the returned reference into a variable called `custMgmt`.
- g. Begin listening for New Customer events by calling `custMgmt.addEventListener("New Customer", listener)`, where `listener` is an object that can respond to New Customer events (see the WebLogic API for a full discussion of event listeners and the `EventListener` interface).
- h. Implement the `onEvent` method of the listener class used in the step above.  
When a New Customer event is received, the `onEvent` method of the listener is called.  
  
The `onEvent` method should then call a method to respond to the event. In this example, the `onEvent` method provides the event object that contains the data associated with the event. The method is called `handleNewCustomer`.
- i. Implement the `handleNewCustomer` method that will respond to the New Customer event.  
  
The `handleNewCustomer` method transforms the XML document in the event to the form expected by the East Coast Order Processing Create Customer service. This transformation may be performed using XSLT or manually using custom transformation code. The end result of the transformation is an XML document that conforms to the schema for the request document of the East Coast Order Processing Create Customer service. Store this document in a variable called `createCustomerRequest`.

## 4 Using Application Views by Writing Custom Code

---

`handleNewCustomer` will then obtain a reference to an instance of the `East Coast.Order Processing Application View` in the same way described for the `East Coast.Customer Management Application View`. This reference is stored into a variable called `orderProc`.

`handleNewCustomer` will then invokes the `Create Customer` service on the `East Coast.Order Processing Application View` by calling `orderProc.invokeService("Create Customer", createCustomerRequest)`. Recall that `createCustomerRequest` is the variable holding the request document for the `Create Customer` service. The response document for this service is stored in a variable named `createCustomerResponse`.

`handleNewCustomer` is finished and returns, leaving itself ready to handle the next incoming `New Customer` event.

When you are finished, a new Java class exists called `SyncCustomerInformation`. This class implements the `Sync Customer Information` business logic. This `SyncCustomerInformation` class uses the `WebLogic` API to get events from the `CRM` system and to invoke services on the `OP` system.

## Example Code for SyncCustomerInformation

Below is the example code for the `SyncCustomerInformation` Java class. It implements the business logic for the scenario described in “About this Example.” Use this example code as a guide for writing your own custom code to implement your enterprise’s business processes.

### Listing 4-1 Full Class Source Code for `SyncCustomerInformation`.

---

```
package com.bea.wlai.test;

import java.util.Hashtable;
import javax.naming.*;
import java.rmi.RemoteException;
import com.bea.wlai.client.*;
import com.bea.wlai.common.*;
import com.bea.document.*;

/**
```

## Steps for Using Application Views by Writing Custom Code

---

```
* This class implements the business logic for the 'Sync Customer Information'
* business process. It uses the WLAI API to listen to events from the CRM
* system, and to invoke services on the OP system. It assumes that there
* are two ApplicationViews defined and deployed in the 'East Coast'
* namespace. The application views and their required events and services
* are shown below.
*
* Customer Management
*   events (New Customer)
*   services (none)
*
* Order Processing
*   events (none)
*   services (Create Customer)
*/
public class SyncCustomerInformation
    implements EventListener
{
    /**
     * Main method to start this application. No args are required.
     */
    public static void
    main(String[] args)
    {
        // Check that we have the information needed to connect to the server.

        if (args.length != 3)
        {
            System.out.println("Usage: SyncCustomerInformation ");
            System.out.println("      <server url> <user id> <password>");
            return;
        }

        try
        {
            // Create an instance of SyncCustomerInformation to work with

            SyncCustomerInformation syncCustInfo =
                new SyncCustomerInformation(args[0], args[1], args[2]);

            // Get a connection to WLAI

            InitialContext initialContext =
                syncCustInfo.getInitialContext();

            // Get a reference to an instance of the 'East Coast.Customer Management'
            // Application View

            ApplicationView custMgmt =
```

## 4 Using Application Views by Writing Custom Code

---

```
        syncCustInfo.getInstanceOfCustomerManagement();

        // Add the listener for 'New Customer' events. In this case we have
        // our application class implement EventListener so it can listen for
        // events directly.

        custMgmt.addEventListener("New Customer", syncCustInfo);

        // Process up to 10 events and then quit.

        syncCustInfo.setMaxEventCount(10);
        syncCustInfo.processEvents();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }

    return;
}

/**
 * EventListener method to respond to 'New Customer' events
 */
public void
onEvent(IEvent newCustomerEvent)
{
    try
    {
        // Print the contents of the incoming 'New Customer' event.

        System.out.println("Handling new customer: ");
        System.out.println(newCustomerEvent.toXML());

        // Handle it

        IDocument response = handleNewCustomer(newCustomerEvent.getPayload());

        // Print the response

        System.out.println("Response: ");
        System.out.println(response.toXML());

        // If we have processed all the events we want to, quit.

        m_eventCount++;
        if (m_eventCount >= m_maxEventCount)
        {
            quit();
        }
    }
}
```



## Steps for Using Application Views by Writing Custom Code

---

```
    }
  }
  catch (Exception e)
  {
    e.printStackTrace();
    System.out.println("Quitting...");
    quit();
  }
}

/**
 * Handles any 'New Customer' event by invoking the 'Create Customer'
 * service on the 'Order Processing' ApplicationView. The response
 * document from the service is returned as the return value of this
 * method.
 */
public IDocument
handleNewCustomer(IDocument newCustomerData)
    throws Exception
{
    // Get an instance of the 'Order Processing' ApplicationView.

    ApplicationView orderProc = getInstanceOfOrderProcessing();

    // Transform the data in newCustomerData to be appropriate for the
    // request document for 'Create Customer' on the 'Order Processing'
    // ApplicationView.

    IDocument createCustomerRequest =
        transformNewCustomerToCreateCustomerRequest(newCustomerData);

    // Invoke the service

    IDocument createCustomerResponse =
        orderProc.invokeService("Create Customer", createCustomerRequest);

    // Return the response

    return createCustomerResponse;
}

// -----
// Member Variables
// -----

/**
 * The url for the WLAI server (e.g. t3://localhost:7001)
 */
private String m_url;
```

## 4 Using Application Views by Writing Custom Code

---

```
/**
 * The user id to use when logging into WLAI.
 */
private String m_userID;

/**
 * The password to use when logging in to WLAI as the user given in
 * m_userID.
 */
private String m_password;

/**
 * The initial context to use when communicating with WLAI
 */
private InitialContext m_initialContext;

/**
 * The NamespaceManager for all namespace operations
 */
private NamespaceManager m_namespaceMgr;

/**
 * The ApplicationViewManager for all ApplicationView operations
 */
private ApplicationViewManager m_appViewMgr;

/**
 * An instance of the 'East Coast.Order Processing' ApplicationView for
 * use in handleNewCustomer.
 */
private ApplicationView m_orderProc;

/**
 * Hold the maximum number of events to be processed in handleNewCustomer
 */
private int m_maxEventCount;

/**
 * Count of the events processed in handleNewCustomer
 */
private int m_eventCount;

/**
 * A monitor variable to enable us to wait until we are asked to quit
 */
private String m_doneMonitor = new String("Done Monitor");

/**
```

## Steps for Using Application Views by Writing Custom Code

---

```
* A flag indicating we are done or not.
*/
private boolean m_done = false;

// -----
// Utility Methods
// -----

/**
 * Constructor.
 */
public SyncCustomerInformation(String url, String userID, String password)
{
    m_url = url;
    m_userID = userID;
    m_password = password;
}

/**
 * Establish an initial context to WLAI.
 */
public InitialContext
getInitialContext()
    throws NamingException
{
    // Set up properties for obtaining an InitialContext to the WLAI server.

    Hashtable props = new Hashtable();

    // Fill in the properties with the WLAI host, port, user id, and password.

    props.put(Context.INITIAL_CONTEXT_FACTORY,
        "weblogic.jndi.WLInitialContextFactory");
    props.put(Context.PROVIDER_URL, m_url);
    props.put(Context.SECURITY_PRINCIPAL, m_userID);
    props.put(Context.SECURITY_CREDENTIALS, m_password);

    // Connect to the WLAI server

    InitialContext initialContext = new InitialContext(props);

    // Store this for later

    m_initialContext = initialContext;

    return initialContext;
}

/**
```

## 4 Using Application Views by Writing Custom Code

---

```
    * Get an instance of the 'East Coast.Customer Management' ApplicationView.
    */
public ApplicationView
getInstanceOfCustomerManagement()
    throws NamingException, NamespaceException,
           ApplicationViewException, RemoteException
{
    // Set up the namespace and ApplicationView manager instances

    setUpManagers();

    // Get a reference to the root namespace

    INamespace root = m_namespaceMgr.getRootNamespace();

    // Get a temporary reference to the 'East Coast' namespace

    NamespaceObjectHandle eastCoastHandle =
        root.getObjectHandle("East Coast");

    // Use the eastCoastHandle to get the East Coast namespace from the
    // NamespaceManager

    INamespace eastCoast = m_namespaceMgr.
        getNamespace(eastCoastHandle.getQualifiedName());

    // Use the eastCoast namespace to get the 'Customer Management'
    // ApplicationView handle

    NamespaceObjectHandle custMgmtHandle =
        eastCoast.getObjectHandle("Customer Management");

    // Use the handle to get the ApplicationView instance from the
    // ApplicationViewManager

    ApplicationView custMgmt =
        m_appViewMgr.
            getApplicationViewInstance(custMgmtHandle.getQualifiedName());

    return custMgmt;
}

/**
 * Get an instance of the 'East Coast.Order Processing' ApplicationView.
 */
public ApplicationView
getInstanceOfOrderProcessing()
    throws NamingException, NamespaceException,
           ApplicationViewException, RemoteException
```

## Steps for Using Application Views by Writing Custom Code

---

```
{
    // Only do this if we don't have a saved one

    if (m_orderProc == null)
    {
        // Set up the namespace and ApplicationView manager instances

        setUpManagers();

        // Get a reference to the root namespace

        INamespace root = m_namespaceMgr.getRootNamespace();

        // Get a temporary reference to the 'East Coast' namespace

        NamespaceObjectHandle eastCoastHandle =
            root.getObjectHandle("East Coast");

        // Use the eastCoastHandle to get the East Coast namespace from the
        // NamespaceManager

        INamespace eastCoast = m_namespaceMgr.
            getNamespace(eastCoastHandle.getQualifiedName());

        // Use the eastCoast namespace to get the 'Customer Management'
        // ApplicationView handle

        NamespaceObjectHandle orderProcHandle =
            eastCoast.getObjectHandle("OrderProcessing");

        // Use the handle to get the ApplicationView instance from the
        // ApplicationViewManager

        ApplicationView orderProc =
            m_appViewMgr.
                getApplicationViewInstance(orderProcHandle.getQualifiedName());

        // Save this instance for use in handleNewCustomer()

        m_orderProc = orderProc;
    }

    return m_orderProc;
}

/**
 * Establish our namespace and ApplicationView manager references.
 */
public void
```

## 4 Using Application Views by Writing Custom Code

---

```
setUpManagers()
    throws NamingException, NamespaceException, ApplicationViewException,
           RemoteException
{
    // Make sure we are connected

    if (m_initialContext == null)
    {
        getInitialContext();
    }

    // Get a NamespaceManager

    if (m_namespaceMgr == null)
    {
        NamespaceManagerHome nmh =
            (NamespaceManagerHome)m_initialContext.
                lookup("com.bea.wlai.client.NamespaceManagerHome");
        m_namespaceMgr = nmh.create();
    }

    // Get an ApplicationViewManager

    if (m_appViewMgr == null)
    {
        m_appViewMgr =
            new ApplicationViewManager(m_initialContext);
    }
}

/**
 * Transform the document in the 'New Customer' event to the document
 * required by the 'Create Customer' service.
 */
public IDocument
transformNewCustomerToCreateCustomerRequest(IDocument newCustomerData)
    throws Exception
{
    // We could do an XSLT transform here, or manually move data from the
    // source to the target document. The details of this transformation
    // are out of the scope of this sample. For information on XSLT see
    // http://www.w3.org/TR/xslt. For more information on manually moving
    // data between documents, see the Javadoc documentation for the
    // com.bea.document.IDocument interface.

    return newCustomerData;
}

/**
```

## Steps for Using Application Views by Writing Custom Code

---

```
* Event processing/wait loop
*/
public void
processEvents()
{
    synchronized(m_doneMonitor)
    {
        while (!m_done)
        {
            try
            {
                m_doneMonitor.wait();
            }
            catch (Exception e)
            {
                // ignore
            }
        }
    }
}

/**
 * Sets the max number of events we want to process.
 */
public void
setMaxEventCount(int maxEventCount)
{
    m_maxEventCount = maxEventCount;
}

/**
 * Method to force this application to exit (cleanly)
 */
public void
quit()
{
    synchronized(m_doneMonitor)
    {
        m_done = true;
        m_doneMonitor.notifyAll();
    }
}
}
```

## **4** Using Application Views by Writing Custom Code

---



# 5 Using the Application View Management Console

This section contains information on the following subjects:

- Before You Begin
- Introduction to Using the Application View Management Console
- Steps for Using the Application View Management Console
  - Logging On to the Application View Management Console
  - Creating Folders
  - Removing Application Views
  - Removing Folders

## Before You Begin

Before you attempt to work with folders, ensure that the following prerequisite is satisfied:

- WebLogic Application Integration is running.

## Introduction to Using the Application View Management Console

Use the Application View Management Console to access, organize, and edit all application views in your enterprise. You can use the Application View Management Console to create new folders and to add new application views to the folders. These folders allow you to organize your application views according to your own navigation scheme, regardless of the adapter the application view belongs to.

## Steps for Using the Application View Management Console

This section explains how to organize application views into folders using the Application View Management Console. The actual folders you set up depend on your organization.

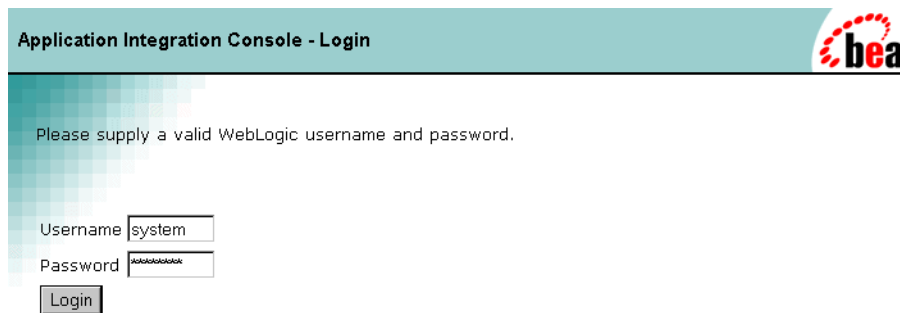
## Logging On to the Application View Management Console

The first step in managing application views is to log on to the Application View Management Console. To log on:

1. Launch a browser window.
2. Open the URL for your system's Application View Management Console. The actual URL you enter depends on your system. It should follow the format:

`http://<yourserver>:<yourport>/wlai`

The logon page displays.



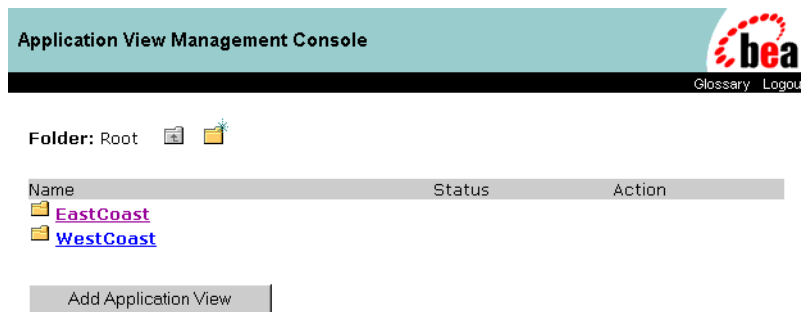
Application Integration Console - Login

Please supply a valid WebLogic username and password.

Username

Password



3. To log on to the Application View Management Console, enter your WebLogic username and password, then click OK. The Application View Management Console displays.



Application View Management Console

Glossary Logout

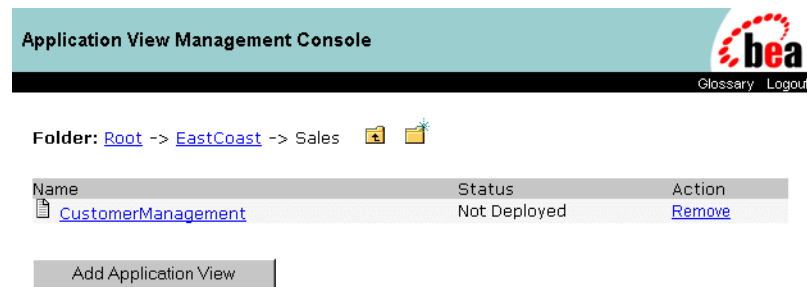
Folder: Root

Name	Status	Action
 EastCoast		
 WestCoast		

### Creating Folders

Create folders to organize the application views in your enterprise. Folders can contain application views and other folders. Once you create a folder, you can not move it to another folder, and you can remove the folder only if it is empty. Once you create an application view in a folder, you can remove the application view, but you can not move it to another folder. To create a folder:

1. While logged on to the Application View Management Console, navigate to the folder where you want to create the new folder.



2. Click the New Folder icon. The Add Folder page displays.



3. In the New Folder field, enter a name.  
Invalid characters: # \ + & ` " . space
4. Click Save.

## Removing Application Views

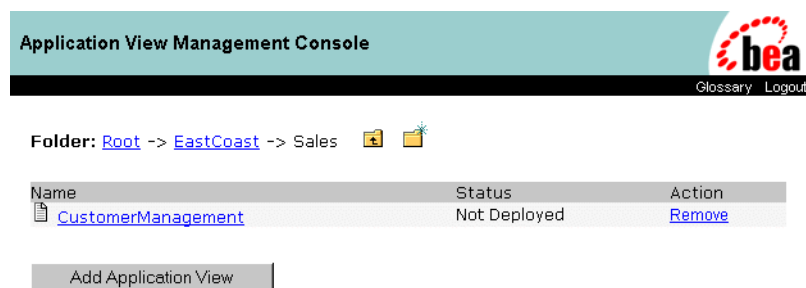
Remove application views when they become obsolete or the application is retired.

You can remove an application view only if the following conditions are true:

- You have undeployed the application view (see “Undeploying an Application View”). That is, the application view status reads “Not Deployed.”
- You are logged on to the WebLogic server using a user account that has the appropriate “write” privileges.

To remove an application view:

1. While logged on to the Application View Management Console, navigate to the folder where the target application view is located.



2. Click Remove to delete the application view.

## Removing Folders

Remove folders that are no longer needed. Before you can remove a folder, you must remove all of its application views and sub-folders. To remove a folder:

1. While logged on to the Application View Management Console, navigate to the folder where the target folder is located.

## 5 Using the Application View Management Console

---

The screenshot shows the 'Application View Management Console' interface. At the top right is the BEA logo and links for 'Glossary' and 'Logout'. Below the header, it displays the current folder path: 'Folder: [Root](#) -> EastCoast'. A table lists application views with columns for Name, Status, and Action. The table contains four entries: 'Sales', 'Marketing', 'AcctsPayable', and 'AcctsReceivable', each with a 'Remove' link. Below the table is a button labeled 'Add Application View'.

Name	Status	Action
<a href="#">Sales</a>		<a href="#">Remove</a>
<a href="#">Marketing</a>		<a href="#">Remove</a>
<a href="#">AcctsPayable</a>		<a href="#">Remove</a>
<a href="#">AcctsReceivable</a>		<a href="#">Remove</a>

2. Click Remove to delete the folder. A confirmation page displays.

The confirmation dialog box features the BEA logo at the top. The main text asks: 'Are you sure you want to remove? 'AcctsReceivable'?'. At the bottom, there are two buttons: 'Confirm' and 'Cancel'.

3. Click Confirm to delete the folder.

---

# Index

## A

- adapter developers 1-8
- adapter users 1-8
- adapters
  - developing 1-8
  - understanding 1-3
- ADK
  - See adapters, developing*
- AI AsyncResponse event 3-11
- AI Event events 3-19
- AI Plug-in
  - AI GetErrorMsg() function 3-14
  - AI GetResponseDocument() function 3-15
  - AI HasError() function 3-14
- AI Start events 3-16
- application view events
  - adding 2-11
  - setting up workflows to wait for 3-19
  - starting workflows using 3-16
  - testing manually 2-27
  - testing using a service 2-24
  - understanding 1-7
- application view folders
  - creating 5-4
  - removing 5-5
  - understanding 1-7
- application view management console
  - logging on to 5-3
  - understanding 1-7
- application view services
  - adding 2-9
  - asynchronous 1-6
  - calling 3-4
  - synchronous 1-7
  - understanding 1-6
- application views
  - adding events to 2-11
  - adding services to 2-9
  - configuring connection parameters 2-7
  - defining 2-2
  - deploying 2-13
  - editing 2-30
  - removing 5-5
  - security 2-16
  - testing events 2-23
  - users of 1-10
  - using by writing custom code 1-11
  - using in WebLogic Process Integrator 1-11
  - when to define 1-4
- applications 1-3
- asynchronous application view services
  - calling from workflows 3-4
  - receiving responses from 3-10
  - understanding 1-6
- AsyncServiceResponse variable
  - in AI GetErrorMsg() 3-14
  - in AI GetResponseDocument() 3-15
  - in AI HasError() 3-14

---

## B

business analysts 1-8  
business processes  
    in workflows 1-11  
    using custom code 1-11

## C

connection parameters 2-7  
custom code  
    for business processes  
        when to use 1-12  
        writing 4-1  
    for defining application views 1-4  
Customer Support ix

## D

documentation  
    how to print viii  
    where to find it viii

## E

e-docs Web site viii  
EIS 1-3  
enterprise information servers  
    *See EIS*  
event nodes  
    receiving service responses 3-10  
    waiting for application view events 3-19  
events  
    *See application view events*

## F

folders  
    *See application view folders*

## J

J2EE Connector Architecture Specification  
    ix  
Java  
    custom coding in 4-1

## M

management console  
    *See application view management  
    console*

## P

Process Integrator  
    *See WebLogic Process Integrator*

## R

Related Information  
    J2EE Connector Architecture  
        Specification ix  
    Sun Microsystems Java site ix  
    WebLogic Server documentation viii  
    XML Schema Specification ix  
Request ID variables  
    when calling services 3-10  
    when receiving service responses 3-13

## S

security 2-16  
services 1-3  
start nodes 3-16  
Sun Microsystems ix  
Sun Microsystems, Inc. Java site ix  
support  
    technical ix  
synchronous application view services  
    calling 3-9  
    understanding 1-7



---

system administrators 1-8

## **T**

Target Fraction parameter 2-16

task nodes 3-4

technical analysts 1-8

## **U**

users

- adapter developers 1-8

- adapter users 1-8

- system administrators 1-8

## **W**

WebLogic Process Integrator

- AI AsyncResponse event 3-11

- using 3-1

- when to use 1-12

- with the AI Plug-in 3-11

- workflows 3-3

WebLogic Server viii

workflows

- using application views in 3-3

## **X**

XML Schema Specification ix