



# BEA WebLogic Adapter for Manugistics™

## User Guide

# Copyright

Copyright © 2003 BEA Systems, Inc. All Rights Reserved.

Copyright © 2003 iWay Software. 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, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

### BEA WebLogic Adapter for Manugistics User Guide

Part Number	Date
N/A	April 2003

---

# Table of Contents

## About This Document

What You Need to Know .....	v
Related Information.....	vi
Contact Us! .....	vi
Documentation Conventions .....	vii

## 1. Introducing the BEA WebLogic Adapter for Manugistics

Introduction .....	1-1
How the BEA WebLogic Adapter for Manugistics Works.....	1-3
How Services in the BEA WebLogic Adapter for Manugistics Work .....	1-3
How Events in the BEA WebLogic Adapter for Manugistics Work .....	1-5
How the Adapter Interfaces with Manugistics Processes .....	1-7

## 2. Creating and Configuring Events

Creating an Application View Folder.....	2-1
Creating an Event Adapter Application View .....	2-3
Configuring an Event Adapter Application View .....	2-5
Deploying an Application View .....	2-10
Testing Events Using Application View Console .....	2-13
Testing Event Adapter Application Views Using WLI Studio .....	2-15

## 3. Creating and Configuring Services

Creating a Service Adapter Application View .....	3-1
Configuring a Service Adapter Application View .....	3-3
Deploying an Application View .....	3-7
Testing Services Using Application View Console .....	3-9
Testing Service Adapter Application Views Using WLI Studio.....	3-11

---

## 4. Creating Schema Repositories

Introduction to Schemas and Repositories .....	4-1
Naming a Schema Repository .....	4-2
The Repository Manifest .....	4-3
Creating a Repository Manifest.....	4-4
Creating a Schema .....	4-5

## 5. Using Tracing

Levels and Categories of Tracing .....	5-2
Tracing and Performance.....	5-3
Creating Traces for Services and Events .....	5-3
Creating Traces for a Service .....	5-4
Creating or Modifying the Tracing Level for an Event.....	5-6
Creating Adapter Logs for an Event.....	5-9

---

# About This Document

The *BEA WebLogic Adapter for Manugistics User Guide* is organized as follows:

- [Chapter 1, “Introducing the BEA WebLogic Adapter for Manugistics,”](#) describes the components of the BEA WebLogic Adapter for Manugistics, describes its features, and presents a high-level description of how the adapter works.
- [Chapter 2, “Creating and Configuring Events,”](#) describes how to create, configure, and test an event adapter.
- [Chapter 3, “Creating and Configuring Services,”](#) describes how to create, configure, and test a service adapter.
- [Chapter 4, “Creating Schema Repositories,”](#) addresses the schema repositories, manifests, and schemas that describe the documents entering and exiting a WebLogic Integration system.
- [Chapter 5, “Using Tracing,”](#) addresses tracing for services and events.

## What You Need to Know

This document is written for system integrators with programming backgrounds and an understanding of Manugistics. Extensive knowledge of Manugistics is not required, but may be helpful in learning about the adapter.

This document provides details on working with the adapter tools to develop online interconnections to Manugistics using BEA WebLogic Integration.

---

# Related Information

The following documents provide additional information for the associated software components:

- *BEA WebLogic Adapter for Manugistics Installation and Configuration Guide*
- *BEA WebLogic Adapter for Manugistics Release Notes*
- BEA WebLogic Server installation and user documentation, which is available at the following URL:  
  
`http://edocs.bea.com/more\_wls.html`
- BEA WebLogic Integration installation and user documentation, which is available at the following URL:  
  
`http://edocs.bea.com/more\_wli.html`

## Contact Us!

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

In your e-mail message, please indicate which version of the adapter documentation you are using.

If you have any questions about this version of the adapter, or if you have problems using it, contact BEA Customer Support through BEA WebSupport at `www.bea.com`. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

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

- Your name, e-mail address, phone number, and fax number
- Your company name and company address

- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

## Documentation Conventions

The following conventions are used throughout this document.

Convention	Item
<b>boldface text</b>	Indicates terms defined in the glossary.
Ctrl+Tab	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> #include <iostream.h> void main ( ) the pointer psz chmod u+w * \tux\data\ap .doc tux.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>

---

Convention	Item
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. <i>Examples:</i> LPT1 SIGNON OR
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
[ ]	Indicates optional items in a syntax line. The brackets themselves should never be typed. <i>Example:</i> buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...
	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.
...	Indicates one of the following in a command line: <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> The ellipsis itself should never be typed. <i>Example:</i> buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...
.	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.



# 1 Introducing the BEA WebLogic Adapter for Manugistics

This section introduces the components of the BEA WebLogic Adapter for Manugistics, describes its features, and presents a high-level description of how the adapter works. It includes the following topics:

- [Introduction](#)
- [How the BEA WebLogic Adapter for Manugistics Works](#)

## Introduction

The corporate world has already realized the value of supply chain management software, but a powerful suite of supply chain management software is only a start. The real challenge is the integration of the software suite with the rest of your enterprise and other organizations.

The BEA WebLogic Adapter for Manugistics enables an organization to fully integrate its Manugistics NetWORKS products with virtually any other legacy mainframe system, DBMS, data warehouse, EDI, B2B, ERP, SCM, CRM or financial application on any platform. High-speed, low-impact, non-intrusive access to and from Manugistics exposes the critical business logic and data contained with Manugistics

for immediate integration with other applications. The BEA WebLogic Adapter for Manugistics uses the Manuba batch process to import data to and export data from the Manugistics database, or universal data model.

The BEA WebLogic Adapter for Manugistics is an off-the-shelf solution that simplifies the process of real-time integration of Manugistics with other systems. Adapter technology removes the need for costly point-to-point computing environments that are often restrictive and provide little flexibility for change.

As the computing environment expands to include legacy systems, ERP packages, SCM packages, CRM packages, EDI networks, B2B exchanges, and the Internet, the cost and complexity of application integration maintenance grows. In some cases, the point-to-point solution cannot be expanded at all, forcing the integration to be re-engineered. By using the BEA WebLogic Adapter for Manugistics in combination with BEA WebLogic Integration, true integration of Manugistics with all your various disparate applications is not only possible, it is very easy.

Whether you are moving information into or out of Manugistics, the BEA WebLogic Adapter for Manugistics provides a proven, easy to use, cost effective architecture that:

- Protects your organization's investment in legacy systems.
- Reduces the cost of your integration projects.
- Provides the infrastructure to manage change in an ever-changing business environment.

Key features of the BEA WebLogic Adapter for Manugistics include:

- Support for bi-directional message interaction with Manugistics.
- Support for service (Manugistics inbound) and event (Manugistics outbound) adapter integration operations with Manugistics presenting XML schemas to WebLogic Integration Studio.
- Full support of XML document transformation to populate XML documents with the results of a Manugistics request, as well as to populate a Manugistics request document with data from external XML documents.

# **How the BEA WebLogic Adapter for Manugistics Works**

The BEA WebLogic Adapter for Manugistics is integrated with the Application Integration (AI) component inside of WebLogic Integration. The adapter provides services and events for the Manugistics supply chain management solutions.

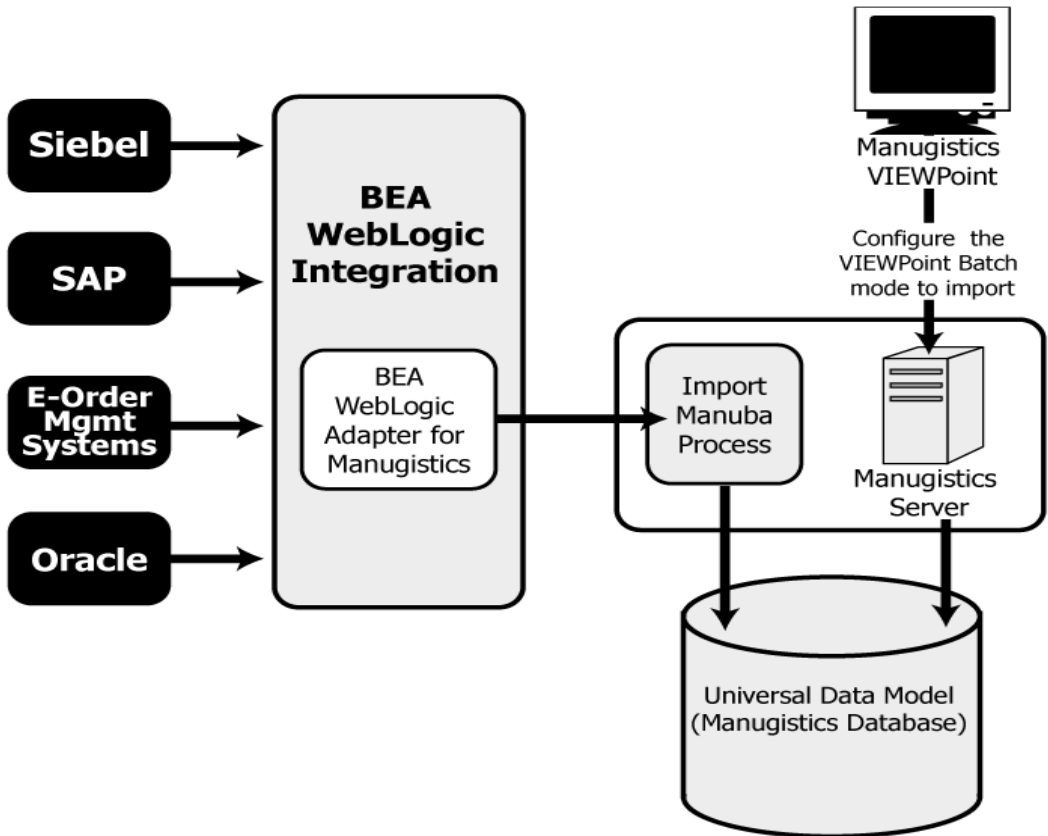
Manugistics NetWORKS provides forecasting solutions that aids in predicting future customer demand, alerting of potential supply problems, finding patterns undetected in demand and helping to manage inventory planning to create time-phased inventory strategies.

The BEA WebLogic Adapter for Manugistics NetWORKS supports import into and export from the Manugistics universal data model (UDM), which is a database that serves as a central storage point for all supply chain storage and optimization data. The adapter interfaces with the UDM through the Manugistics Manuba batch process. Manugistics recommends import into and export from the Manugistics database using the Manuba process to ensure data integrity.

## **How Services in the BEA WebLogic Adapter for Manugistics Work**

Services configured in the BEA WebLogic Adapter for Manugistics allow you to use data from order processing, enterprise resource planning (ERP), and enterprise-wide data warehouse systems to update the Manugistics UDM. Relevant data for Manugistics includes order and sales information, purchase orders, inventory balances, and forecasts usually available in the external systems. The adapter uses the Manugistics Manuba process to update the Manugistics database. The following illustration depicts the service architecture.

**Figure 1-1 BEA WebLogic Adapter for Manugistics Service Architecture**



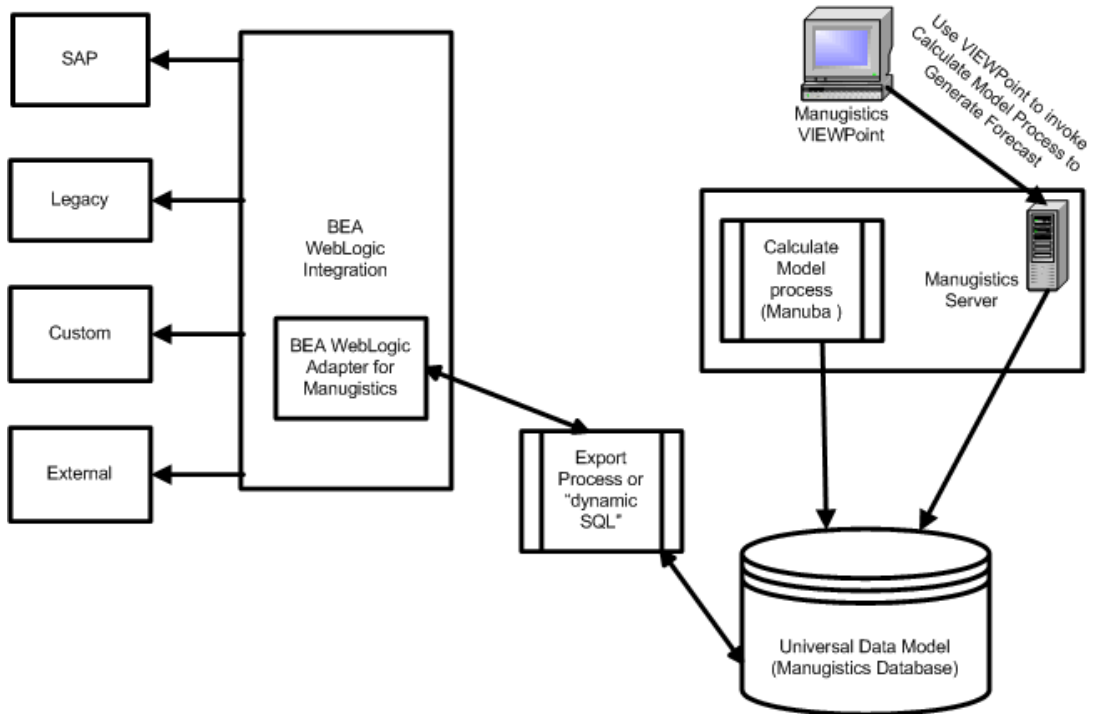
As data becomes available in applications such as Oracle, Siebel, or SAP, and is passed to the adapter within the BEA WebLogic Integration environment, it is parsed by the adapter to ensure its compatibility with the Manugistics UDM. Use the Manugistics VIEWpoint client, a graphical user interface for the Manugistics platform, to configure a batch mode import file to match the data passed from the adapter. For more information on the how the adapter works with Manugistics, see [“How the Adapter Interfaces with Manugistics Processes”](#) on page 1-7.

Manugistics provides the Manuba batch process for importing data into and exporting data from the Manugistics UDM. The adapter invokes that Manuba process automatically. Once the configuration file for the batch import process is configured, the adapter reads the configuration file and creates a new file with the required format, path, and file name. After Manuba is invoked, the data is imported into the UDM.

## **How Events in the BEA WebLogic Adapter for Manugistics Work**

Events configured in the BEA WebLogic Adapter for Manugistics allow you to use forecast data generated by Manugistics to update order processing, enterprise resource planning (ERP), and enterprise-wide data warehouse systems. Relevant data for the Manugistics NetWORKS solutions includes order and sales information, purchase orders, inventory balances, and forecasts usually available in the external systems. The adapter uses the Manuba process to export data from the Manugistics UDM. The following illustration depicts the service architecture.

**Figure 1-2 BEA WebLogic Adapter for Manugistics Event Architecture**



In this event architecture, which is an example showing the calculate model in the Manugistics Demand Planner, you invoke the calculate model process to generate forecast data using either the Manuba calculate model process or the VIEWPoint interface. The BEA WebLogic Adapter for Manugistics event polls the UDM. Once the UDM is updated with the forecast data, the adapter converts the data into XML and passes it through WebLogic Integration so that it can be used to update systems such as SAP and Siebel. For more information on the how the adapter works with Manugistics, see [“How the Adapter Interfaces with Manugistics Processes” on page 1-7.](#)

## How the Adapter Interfaces with Manugistics Processes

The BEA WebLogic Adapter for Manugistics relies on the Manugistics Manuba batch process to import data to and export data from the UDM to ensure the integrity of the data. The Manuba batch process requires a configuration file before it can be invoked by the adapter.

**Note:** You must create the Manugistics configuration (.lst) file for use by the Manuba process before you configure services and events. See your Manugistics documentation for more information on creating configuration files.

The configuration file contains information such as the userview. In VIEWpoint, userviews are objects that display data and allow you to interact with the system. Any time you work with data in a database table or in a what-if simulation, you are working with a userview. In addition, the configuration file contains information about data selection and process.

For example, a typical export process is as follows:

```
manuba fcst_export.lst fcst_export.log
```

The configuration file `fcst_export.lst` has the following components:

```
DESKTOP DEMANDPLANNING  
SELECTION FCST_TYPE_1  
USERVIEW FCST_EXPORT.
```

The `USERVIEW FCST_EXPORT` is in the Demand database and has all the format patterns of the FCST table, the location of the file to be exported, and the file name. The Manuba batch process uses this information to carry out the export of data from the Manugistics database.

The BEA WebLogic Adapter for Manugistics receives data from systems such as SAP and transforms the data into XML. Once it is in XML, the adapter can convert it to the format that can be used by the Manugistics Manuba process to update the Manugistics UDM. For events, the adapter can poll the Manugistics UDM and convert the data to the appropriate XML format. It is then passed from WebLogic integration to update external systems. The adapter converts the XML into a format required by a particular ERP or legacy system, if necessary.

# **1**    *Introducing the BEA WebLogic Adapter for Manugistics*

---

When you configure services, you must provide information about your Manugistics system, such as the location of the Manuba batch file, that will allow the adapter to invoke the Manuba process.



# 2 Creating and Configuring Events

An event adapter is the inbound interface from a Manugistics application to WebLogic Integration Studio. This section describes how to create, configure, and test an event adapter. It contains the following topics:

- [Creating an Application View Folder](#)
- [Creating an Event Adapter Application View](#)
- [Configuring an Event Adapter Application View](#)
- [Deploying an Application View](#)
- [Testing Events Using Application View Console](#)
- [Testing Event Adapter Application Views Using WLI Studio](#)

## Creating an Application View Folder

Application views reside within WebLogic Integration. WebLogic Integration provides you with a root folder in which you can store all of your application views. You can create additional folders to organize related application views into groups.

To create an application view folder:

1. Log on to the WebLogic Integration Application View Console at `http://appserver-host:port/wlai`

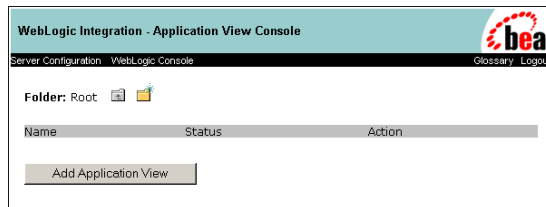
## 2 Creating and Configuring Events

---

Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

**Note:** If the user name is not *system*, it must be included in the adapter group. For more information on adding the administrative server user name to the adapter group, see the *BEA WebLogic Adapter for Manugistics Installation and Configuration Guide*.

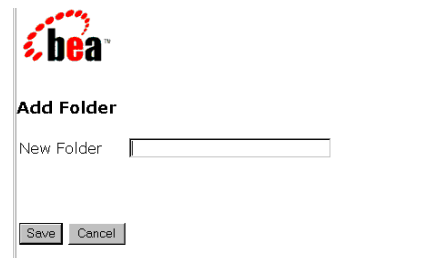
**Figure 2-1 Application View Console Main Window**



2. Click the new folder icon.

The Add Folder window opens.

**Figure 2-2 Figure 2-2 Add Folder Window**



- a. Enter the folder name.
- b. Click Save.

You have finished creating the application view folder.

# Creating an Event Adapter Application View

To create an event adapter application view, follow the steps in this section.

**Note:** You must create a schema repository and create schemas before you create an event adapter application view. For more information, see [Chapter 4, “Creating Schema Repositories.”](#)

To create an event adapter application view:

1. Log on to the WebLogic Integration Application View Console at `http://appserver-host:port/wlai`

Here *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

2. Select the desired application view folder.
3. Click Add Application View.

The Define New Application View window opens.

**Figure 2-3 Application View Console: Define New Application View**

**Define New Application View**

Server Configuration WebLogic Console Glossary Logout

This page allows you to define a new application view

Folder: [Root](#)

Application View Name: \*

Description:

Associated Adapter: -- None --

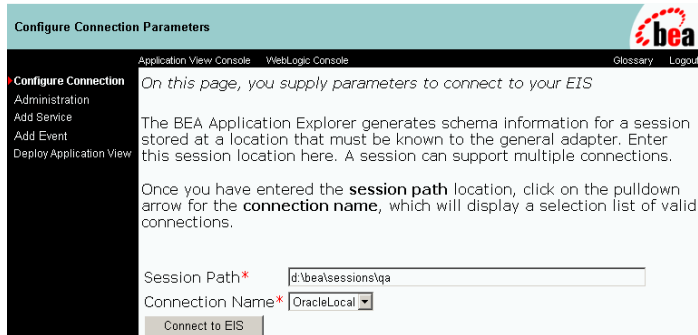
OK Cancel

- a. Enter a name and description for the application view.
- b. Select `BEA_MGISTICS_1_0` from the Associated Adapter drop-down list.

- c. Click OK.

The Configure Connection Parameters window opens.

**Figure 2-4 Application View Console: Configure Connection Parameters**



**Note:** You can access the Configure Connection Parameters window when the application view is not deployed by selecting the Reconfigure connection parameters link. If the application view is deployed, you can access the window by first undeploying the application view.

- a. Type the name of the BEA WebLogic Adapter for Manugistics session base directory in the Session path field.

This directory holds your Manugistics schema information and contains the subdirectory *Manugistics/YourConnectionName*.

For example, the session base directory might be

d:\bea\bse\sessions\default, with the schema repository, containing a repository manifest and schemas, residing in

d:\bea\bse\sessions\default\Manugistics\ManugisticsProd. For more information about schema repositories, see [Chapter 4, “Creating Schema Repositories.”](#)

- b. Select the session name, also known as the connection name, from the Connection name drop-down list.
- c. Click Connect to EIS.

The Application View Administration window opens.

**Figure 2-5 Application View Administration Window**

The screenshot shows the 'Application View Administration for Manu' window. The left sidebar has a menu with 'Administration' selected. The main content area has a header with 'Application View Console' and 'WebLogic Console' tabs, and a 'Glossary' and 'Logout' link. Below the header is a description: 'This page allows you to add events and/or services to an application view.' The 'Description' field shows 'No description available for Manu.' with an 'Edit' link. Below this is a 'Connection Criteria' section with a table of values: Connection Name: OracleLocal, nOT\_VALID\_000: true, Additional Log Category: Manu, Log Level: DEBUG, Root Log Category: BEA\_MGISTICS\_1\_0, Session Path: d:\bea\sessions\qa, Log Configuration File: BEA\_MGISTICS\_1\_0.xml, and Message Bundle Base: BEA\_MGISTICS\_1\_0. Below the table is a link 'Reconfigure connection parameters for Manu'. There are two sections, 'Events' and 'Services', each with an 'Add' button. At the bottom is a 'Save' button and a lightbulb icon.

Connection Criteria	
Connection Name:	OracleLocal
nOT_VALID_000:	true
Additional Log Category:	Manu
Log Level:	DEBUG
Root Log Category:	BEA_MGISTICS_1_0
Session Path:	d:\bea\sessions\qa
Log Configuration File:	BEA_MGISTICS_1_0.xml
Message Bundle Base:	BEA_MGISTICS_1_0

4. Click Save.

You have finished creating the application view for the event adapter.

Note that you must add an event, as described in “[Configuring an Event Adapter Application View](#),” before you are able to deploy the application view.

## Configuring an Event Adapter Application View

An event adapter application view contains all events that are expected to arrive at an instance of the event adapter. You can add many events to an application view. Each event has a schema for the arriving message (a message is also known as a document). A service should be added for each event that is used by the application view.

To add an event to, and deploy, an event adapter application view:

1. Log on to the WebLogic Integration Application View Console at `http://appserver-host:port/wlai`.

## 2 Creating and Configuring Events

2. Select the folder in which this application view resides.
3. Select the application view.
4. In the Administration window of the WebLogic Integration Application View Console, choose Add Event.

The Add Event window opens.

**Figure 2-6 Add Event Window - ManugisticsSQL**

**Add Event**

Application View Console 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:\*

**ManugisticsSQL**

Character Set Encoding*	UTF-8
Driver*	oracle.jdbc.driver.OracleDriver
url*	jdbc:oracle:thin:@localhost:1521:ms
User Name	jstec
Password	*****
Format*	field
Maximum Rows	5000
SQL Post Query	
Delete Keys	
Polling Interval	180
Data Source Name	

schema: sql0000\_fcstdata

**settings**

Trace on/off	<input type="checkbox"/>
Verbose Trace on/off	<input type="checkbox"/>
Document Trace on/off	<input type="checkbox"/>

Add

5. Specify the Manugistics event's properties, described in the following table.

**Table 2-1 Event Properties**

Parameter	Description
Character Set Encoding* (*Required)	Encoding of the data being read.

**Table 2-1 Event Properties**

Parameter	Description
Driver* (*Required)	Vendor-specific JDBC driver for access to the database. This parameter requires a fully-qualified name.
url* (*Required)	<p>A database URL (or JDBC URL) is a platform-independent way of addressing a database. A database/JDBC URL has the following form:</p> <pre>jdbc:[subprotocol]:[node]/[databaseName]</pre>
User Name	Valid user name for access to the database.
Password	Valid password associated with the user name for access to the database.
Format * (*Required)	<p>Choose one of the following:</p> <ul style="list-style-type: none"> <li>■ <b>column.</b> The data produced is returned field by field, and each field is enclosed in &lt;column&gt; tags. The column tag has an attribute whose value is the name of the field; for example, <pre>&lt;row&gt; &lt;column name="ID"&gt;1000&lt;/column&gt; &lt;column name="First_Name"&gt;Scott&lt;/column&gt; &lt;/row&gt;</pre> </li> <li>■ <b>field.</b> The data produced is returned field by field, and each field is enclosed in a tag that bears the field name; for example, <pre>&lt;row&gt; &lt;ID&gt;1000&lt;/column&gt; &lt;FIRST_NAME&gt;Scott&lt;/column&gt; &lt;/row&gt;</pre> </li> </ul>
Maximum Rows	<p>Number of data rows to be retrieved from the database table in a single operation. For example, if five were specified, then up to five rows are read and processed in a single operation. In most circumstances, you should not allow this parameter to exceed the number of parallel threads available for execution.</p> <p>The number of events created in a single polling interval is dependent on the setting for Maximum Rows, and the number of new rows added to the database since the last time the database was polled. For example, if Maximum Rows is set to 5, and 23 new rows are found to have been added when the database is polled, four events containing five rows and one event containing three rows are created.</p>

## 2 Creating and Configuring Events

**Table 2-1 Event Properties**

Parameter	Description
SQL Post-Query	<p>SQL Query that is executed after the initial query request.</p> <p>If this parameter is not configured, the following command is executed:</p> <pre>DELETE field1,field2... from table_name</pre> <p>This parameter should not be configured if the RDBMS event adapter exit is configured.</p> <p>Two types of operators are available: ?fieldname and ^fieldname.</p> <ul style="list-style-type: none"><li>■ The ?fieldname will evaluate at run time to ?fieldname= value.</li><li>■ The ^fieldname will evaluate at run time to value.</li></ul> <p>A SQL Post query using the ? can be used in an update statement as follows: update <i>tablename</i> where ?fieldname.</p> <p>For example, update stock_prices_temp where ?RIC.</p> <p>A SQL Post Query using the ^ can be used in an insert statement as follows:</p> <p>Insert into <i>tablename</i> values (^fieldname1, ^fieldname2, ^fieldname3).</p> <p>For example, Insert into stock_prices_temp values (^RIC, ^PRICE, ^UPDATED).</p>
Delete Keys	<p>Comma separated list of keys used in the DELETE statement. A delete operates on keys, so you should enter the table's key columns in this parameter.</p>
Polling Interval	<p>Interval in seconds at which the database is monitored for new rows. If this parameter is not configured, the default value is two seconds.</p>
Data Source Name	<p>The Data Source JNDI name for the JDBC connection pool to use for connecting to the RDBMS system. If a value is present, the adapter will use the connection pool to connect to the RDBMS. If no value is specified, connection will use the Driver, URL, UserId and Password specified in the service.</p>

6. Select a schema from the drop-down list.
7. Select trace settings as follows:



**Table 2-2 Trace Options**

Parameter	Description
Trace	Generates a basic trace that displays the input XML (up to 300 bytes) before parsing, and shows the request being processed. For more information about tracing, see <a href="#">Chapter 5, “Using Tracing.”</a>
Verbose Trace	Generates a trace that displays configuration parameters used by the adapter. For more information about tracing, see <a href="#">Chapter 5, “Using Tracing.”</a>
Document Trace	Generates a trace that displays the input document after it was analyzed and the response document being returned. For more information about tracing, see <a href="#">Chapter 5, “Using Tracing.”</a>

## 8. Click Add.

The Application View Administration window opens.

**Figure 2-7 Application View Administration Window**

The screenshot shows the 'Application View Administration for Manu' window. The left sidebar contains a navigation menu with 'Administration' selected. The main content area displays the configuration for the 'ManuSQL' connection. The 'Description' field is empty, with a note that no description is available for 'Manu'. Below this, the 'Connection Criteria' section lists various parameters such as 'Connection Name', 'nOT\_VALID\_000', 'Additional Log Category', 'Log Level', 'Root Log Category', 'Session Path', 'Log Configuration File', and 'Message Bundle Base'. At the bottom, there are sections for 'Events' and 'Services', each with an 'Add' button. The 'Services' section lists 'ManuService' with links for 'Edit', 'Remove Service', 'View Summary', 'View Request Schema', and 'View Response Schema'.

9. Click Save.

# Deploying an Application View

You can deploy an application view when you have added at least one event or service to it. You must deploy an application view before you can test its services and events or use the application view in the WebLogic Server environment. Application view deployment places relevant metadata about its services and events into a run-time metadata repository. Deployment makes the application view available to other WebLogic Server clients. This means business processes can interact with the application view, and you can test the application view's services and events.

After you configure an event or service, you can deploy your application view from the Application View Console Administration window.

1. If it is not already open, open the application view administration window of the application to be deployed. For more information, see “Editing an Application View” in “Defining an Application View” in *Using Application Integration*:

See <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

The Application Administration window appears.

**Figure 2-8 Application Administration Window**

**Application View Administration for Manu**

Application View Console 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:** No description available for Manu. [\\_Edit](#)

**Connection Criteria**

Connection Name:	OracleLocal
nOT_VALID_000:	true
Additional Log Category:	Manu
Log Level:	DEBUG
Root Log Category:	BEA_MGISTICS_1_0
Session Path:	d:\bea\sessions\qa
Log Configuration File:	BEA_MGISTICS_1_0.xml
Message Bundle Base:	BEA_MGISTICS_1_0

[Reconfigure connection parameters for Manu](#)

**Events** [Add](#)

**ManuSQL** [Edit](#) [Remove Event](#) [View Summary](#) [View Event Schema](#)

**Services** [Add](#)

**ManuService** [Edit](#) [Remove Service](#) [View Summary](#) [View Request Schema](#) [View Response Schema](#)

[Continue](#) [Save](#)

2. Click Continue.

The Deploy Application View window opens.

**Figure 2-9 Deploy Application View Window**

**Deploy Application View Manu to Server**

Application View Console 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.

**Required Service Parameters**

Enable asynchronous service invocation? ☒

**Required Event Parameters**

Event Router URL \*

**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 Manu using J2EE Security](#)

Deploy ☒ Deploy persistently? ☐ Save

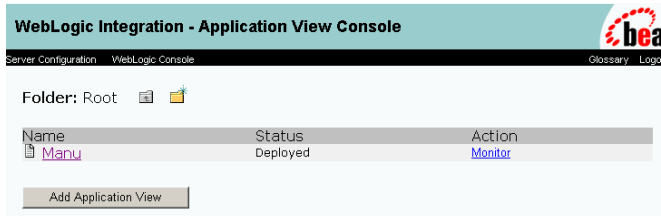
3. Modify event parameters, connection pool parameters, log configuration, and security as required. For more information, see “Defining an Application View” in *Using Application Integration*:

See <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

4. Click Deploy to deploy the event adapter.
5. To validate that the application view was successfully deployed, proceed to the main Application View Console window and double-click the folder in which you created the application view.

You should see the name of the new application view with a status of deployed.

**Figure 2-10 New Application View Window**



You have finished configuring the event adapter application view. Confirm that you configured it correctly and that it can successfully receive events using the instructions in [“Testing Events Using Application View Console” on page 2-13](#) and [“Testing Event Adapter Application Views Using WLI Studio” on page 2-15](#).

# Testing Events Using Application View Console

You can test configuration using the Application View Console. To confirm that a deployed event adapter application view is correctly configured and can receive events:

1. Log on to the WebLogic Integration Application View Console at `http://appserver-host:port/wlai`.
2. Select the folder in which the application view resides and then select the application view.

The Summary window opens.

**Figure 2-11 Summary Window**

Summary for Application View Manu

Application View Console Server Configuration WebLogic Console Glossary Logout

**Summary**

This page shows the events and services defined for the **Manu** Application View.

Name: Manu  
Description:  
Status: Deployed  
Available Actions: [Undeploy](#)

Connection Security Deploy **Events and Services**

**Events**  
Event Router URL: [http://localhost:7001/BEA\\_MGTISTICS\\_1\\_0\\_EventRouter/EventRouter](http://localhost:7001/BEA_MGTISTICS_1_0_EventRouter/EventRouter)

**ManuSQL** [Test](#) [View Summary](#) [View Event Schema](#)

**Services**

**ManuService** [Test](#) [View Summary](#) [View Request Schema](#) [View Response Schema](#)

3. Click Test for one of the application view's events.

The Test Event window opens.

**Figure 2-12 Test Event Window**

Test Event: ManuSQL

Application View Console WebLogic Console Glossary Logout

**Summary**

This page allows you to test an event. You may create the event by invoking a service, or by manually creating the event.

If you want to use a service invocation to create an event, select the Service option below, and select the service to invoke. Optionally, 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 to create an insert event).

How do you want to create the event?

☐ Service ☒ Manual

How long should we wait to receive the event?  
Time (in milliseconds):

[Test](#)

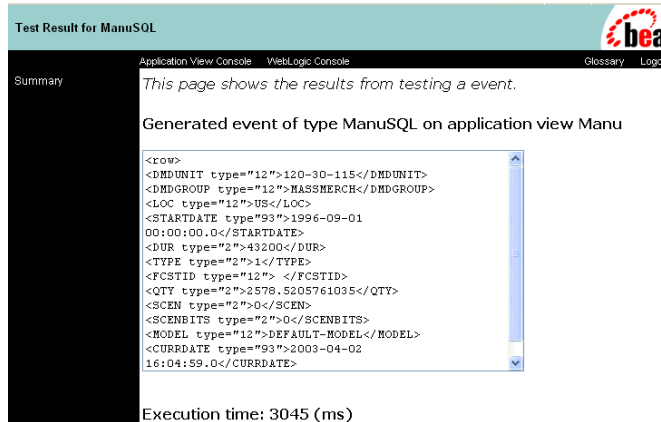
4. Enter 3000 (or a higher value) in the Time field.

This provides a 30-second period during which you can access the Manugistics client program (or your favorite utility) to manually invoke a request from Manugistics to your event adapter.

5. Click Test.

The test output area is displayed.

**Figure 2-13 Test Result Window**



If you wait longer than a minute and do not receive the event's result, assume that the event adapter application view is configured incorrectly. Examine the WebLogic Server Log for information about the event's activity.

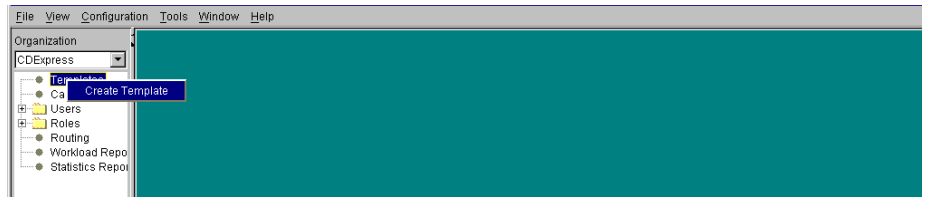
Otherwise, you have confirmed that the event adapter application view is correctly configured and can receive events.

# Testing Event Adapter Application Views Using WLI Studio

You can test configuration using WebLogic Integration Studio. To confirm that a deployed event adapter application view is correctly configured and can receive events:

1. Log on to WebLogic Integration Studio.

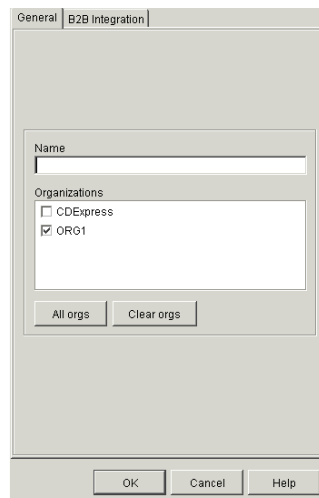
**Figure 2-14** Selecting Create Template in WebLogic Integration Studio



2. In the Organization pane, select an organization to create a new workflow template.
3. Right-click Templates and select Create Template.

The Template Properties window opens.

**Figure 2-15** Template Properties Window

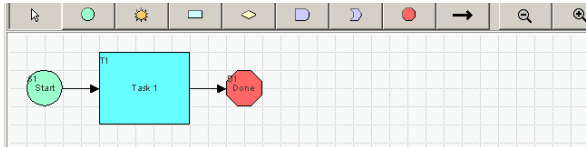


4. Enter a name for your workflow and click OK.
5. Right-click the new template and select Create Template Definition.
6. Click OK.

The template appears in WebLogic Integration Studio.

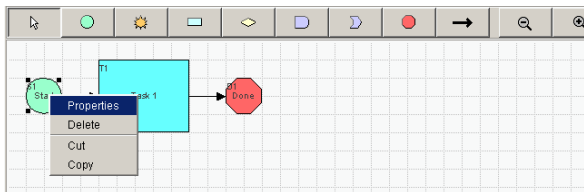


**Figure 2-16** Displaying the New Template in WebLogic Integration Studio



7. To select node properties, right-click the Start node.

**Figure 2-17** Selecting Node Properties in WebLogic Integration Studio



8. Select Properties from the pop-up menu.

The Start Properties window opens.

**Figure 2-18** Start Properties Window

Variable	Expression
----------	------------

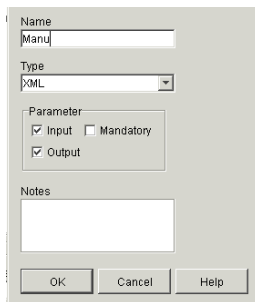
## 2 *Creating and Configuring Events*

---

- a. Choose Event, and then choose AI Start from the drop-down box to the right.
- b. In the event explorer in the left pane, browse the application view folders and select the application view that corresponds to the event adapter.
- c. Open the event adapter and select the desired event.
- d. Select <new> from the Event Document Variable drop-down box.

The Variable Properties window opens.

**Figure 2-19 Variable Properties Window**

The image shows a 'Variable Properties' dialog box. It has a 'Name' field with 'Manu' entered. Below it is a 'Type' dropdown menu showing '<XML>'. Under the 'Parameter' section, there are three checkboxes: 'Input' (checked), 'Mandatory' (unchecked), and 'Output' (checked). At the bottom is a 'Notes' text area. At the very bottom are three buttons: 'OK', 'Cancel', and 'Help'.

- e. Enter a name for the new variable.
  - f. Select the variable type XML from the drop-down list.
  - g. Select the Input and Output options in the Parameter group.
  - h. Click OK.
9. Right-click the template in the organization pane of WebLogic Integration Studio and select Save.
  10. Right-click the event definition folder and select Properties.

The Template Definition window appears.

**Figure 2-20 Template Definition Window**

General | Exception Handlers | B2B Integration

Workflow Label:  ABC

☐ Active

Effective:  Apr 2, 2003 Expiry:  Dec 31, 2003

☐ Enable auditing

Notes:

OK Cancel Help

- a. Ensure that Active is checked.
- b. Click OK.

You may now initiate events from your Enterprise Information System (EIS).

For the BEA WebLogic Adapter for Manugistics, you can create events through Manugistics Manuba or the Manugistics Client (ViewPoint) test program.

11. Return to WebLogic Integration Studio.

12. Right-click the event definition folder and choose Instances.

The Workflow Instances for your event definition appear. You can now track execution of your workflow.

13. Select Started and click Refresh. You should see a list of started workflows:

**Figure 2-21 Workflow Instances Window**

☒ Started ☐ Completed From:  Apr 2, 2003 To:  Apr 2, 2003 Refresh 1 item(s)

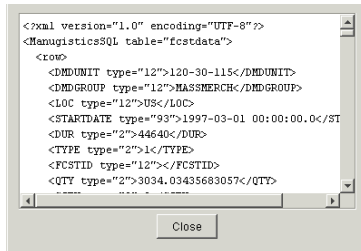
Workflow Label	Started	Completed	Comment
	2003-04-02 16:16:33.441		

14. Right-click any instance of the workflow and select Variables.

The Workflow Variables window opens.

15. Click View XML to see the entire contents of the workflow message (messages are also known as documents).

**Figure 2-22 Workflow Variables Window**



# 3 Creating and Configuring Services

The service adapter for Manugistics is WebLogic Integration's interface to Manugistics. It enables your business processes to move information to your Manugistics application system. This section describes how to create, configure, and test a service adapter application view. It includes the following topics:

- [Creating a Service Adapter Application View](#)
- [Configuring a Service Adapter Application View](#)
- [Deploying an Application View](#)
- [Testing Services Using Application View Console](#)
- [Testing Service Adapter Application Views Using WLI Studio](#)

## Creating a Service Adapter Application View

**Note:** You must create a schema repository and create schemas before you create a service adapter application view. For more information, see [Chapter 4, "Creating Schema Repositories."](#)

To create a service adapter application view.

1. In the Application View Console main window, click Add Application View.  
The Define New Application View window opens.

**Figure 3-1 Define New Application View Window**

**Define New Application View**

Server Configuration WebLogic Console

This page allows you to define a new application view

Folder: [Root](#)

Application View Name: \*

Description:

Associated Adapter: -- None --

OK Cancel

- Enter a name and description for the application view.
- Select BEA\_MGISTICS\_1\_0 from the Associated Adapter list.
- Click OK.

The Configure Connection Parameters window opens.

**Figure 3-2 Configure Connection Parameters Window**

**Configure Connection Parameters**

Application View Console WebLogic Console

On this page, you supply parameters to connect to your EIS

The BEA Application Explorer generates schema information for a session stored at a location that must be known to the general adapter. Enter this session location here. A session can support multiple connections.

Once you have entered the **session path** location, click on the pulldown arrow for the **connection name**, which will display a selection list of valid connections.

Session Path \*

Connection Name \*

Connect to EIS

- Enter the name of the BEA WebLogic Adapter for Manugistics session base directory in the Session path field.

This directory holds your Manugistics schema information and contains the subdirectory Manugistics/*YourConnectionName*.

For example, the session base directory might be

d:\bea\bse\sessions\default, with the schema repository—containing

a repository manifest and schemas—residing in

d:\bea\bse\sessions\default\Manugistics\ManugisticsLab.

- b. Select the session name—also known as the connection name—from the Connection name drop-down list.
- c. Click Connect to EIS.

The Application View Administration window opens.

**Note:** You can access the Configure Connection Parameters window when the application view is not deployed by selecting the Reconfigure connection parameters link. If the application view is deployed, you can access the window by first undeploying the application view.

# Configuring a Service Adapter Application View

After you create the service adapter application view, you must configure it

To configure a service adapter application view:

1. Log on to the Application View Console at

`http://appserver-host:port/wlai`

Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

The main window of the Application View Console opens.

2. Select the folder in which this application view resides, then select the application view.

The Administration window opens.

**Figure 3-3 Administration Window**

The screenshot shows the 'Administration' window for 'Manu'. The left navigation pane has 'Administration' selected. The main content area has a title bar 'Application View Administration for Manu' and a sub-header 'Application View Console WebLogic Console'. Below the sub-header is a description: 'This page allows you to add events and/or services to an application view.' The main content area is divided into sections: 'Description:' with a link to 'Edit', 'Connection Criteria' with a table of values, 'Events' with an 'Add' button, and 'Services' with an 'Add' button. At the bottom is a 'Save' button.

Connection Criteria	
Connection Name:	OracleLocal
nOT_VALID_000:	true
Additional Log Category:	Manu
Log Level:	DEBUG
Root Log Category:	BEA_MGISTICS_1_0
Session Path:	d:\bea\sessions\qa
Log Configuration File:	BEA_MGISTICS_1_0.xml
Message Bundle Base:	BEA_MGISTICS_1_0

3. From the left navigation pane, select Add Service.

The Add Service window opens.

**Figure 3-4 Add Service Window**

The screenshot shows the 'Add Service' window. The left navigation pane has 'Add Service' selected. The main content area has a title bar 'Add Service' and a sub-header 'Application View Console WebLogic Console'. Below the sub-header is a description: 'On this page, you add services to your application view.' The main content area is divided into sections: 'Unique Service Name:\*' with a text input field, 'Manugistics' with a table of values, 'schema:' with a dropdown menu, and 'settings' with three checkboxes. At the bottom is an 'Add' button.

Manugistics	
Drive	~Drive~
ManubaPath*	~ManubaPath~
BatchPath*	~BatchPath~
LSTFile*	~LSTFile~
LOGFile*	~LOGFile~
OutputFile*	~OutputFile~
encoding*	UTF-8



- a. Update the required properties of Manugistics.

These settings correspond to the Manugistics settings that the service adapter uses to communicate with Manugistics.

**Table 3-1 Manugistics Property Settings**

Parameter	Description
Drive	The name of the drive on which Manugistics is running (optional).
ManubaPath* (*Required)	Path to the Manuba process. For example, c:\manu\manuv61\scpo\bin.
BatchPath* (*Required)	Batch process that sets the Manuba environment and invokes it.
LSTFile* (*Required)	The configuration file required by Manuba. For example, Manuba <LSTFILE> <LOGFILE>
LOGFile* (*Required)	The name of the output log file required for Manuba.
OutputFile * (*Required)	The name of the file containing the formatted data. This file is configured in the Manugistics ViewPoint client for the import process.
encoding	Character encoding of the data. The default is UTF-8

4. Select a schema from the drop-down list.
5. Select trace settings, as follows:

**Table 3-2 Trace Options**

Parameter	Description
Trace	Generates a basic trace that displays the input XML (up to 300 bytes) before parsing, and shows the request being processed. For more information about tracing, see <a href="#">Chapter 5, “Using Tracing.”</a>

**Table 3-2 Trace Options**

Parameter	Description
Verbose Trace	Generates a trace that displays configuration parameters used by the adapter. For more information about tracing, see <a href="#">Chapter 5, “Using Tracing.”</a>
Document Trace	Generates a trace that displays the input document after it was analyzed and the response document being returned. For more information about tracing, see <a href="#">Chapter 5, “Using Tracing.”</a>

6. Click Add.

The Administration window opens.

**Figure 3-5 Application View Administration Window**

Application View Administration for Manu

Application View Console 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: No description available for Manu. [\\_Edit](#)

Connection Criteria

Connection Name:	OracleLocal
nOT_VALID_000:	true
Additional Log Category:	Manu
Log Level:	DEBUG
Root Log Category:	BEA_MGISTICS_1_0
Session Path:	d:\bea\sessions\qa
Log Configuration File:	BEA_MGISTICS_1_0.xml
Message Bundle Base:	BEA_MGISTICS_1_0

[Reconfigure connection parameters for Manu](#)

Events [Add](#)

ManuSQL [Edit](#) [Remove Event](#) [View Summary](#) [View Event Schema](#)

Services [Add](#)

ManuService [Edit](#) [Remove Service](#) [View Summary](#) [View Request Schema](#) [View Response Schema](#)

[Continue](#) [Save](#) ?

7. Click Save.

# Deploying an Application View

You can deploy an application view when you have added at least one event or service to it. You must deploy an application view before you can test its services and events or use the application view in the WebLogic Server environment. Application view deployment places relevant metadata about its services and events into a run-time metadata repository. Deployment makes the application view available to other WebLogic Server clients. This means business processes can interact with the application view, and you can test the application view's services and events.

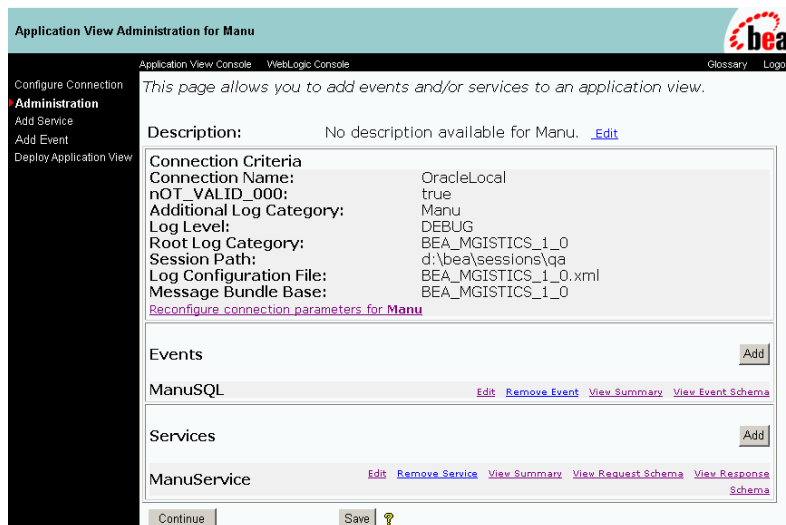
After you configure an event or service, you can deploy your application view from the Application View Console Administration window.

1. If it is not already open, open the application view administration window of the application to be deployed. For more information, see “Editing an Application View” in “Defining an Application View” in *Using Application Integration*:

- See <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

The Application Administration windows opens.

**Figure 3-6 Application View Administration Window**



2. Click Continue.

### 3 Creating and Configuring Services

The Deploy Application View window opens.

**Figure 3-7 Deploy Application View Window**

The screenshot shows the 'Deploy Application View Manu to Server' window. The left sidebar contains a navigation menu with options: 'Configure Connection', 'Administration', 'Add Service', 'Add Event', and 'Deploy Application View' (which is highlighted). The main content area has a title bar with 'Application View Console' and 'WebLogic Console'. Below the title bar, there is a message: 'On this page you deploy your application view to the application server.' The main content area is divided into several sections: 'Required Service Parameters' with a checkbox for 'Enable asynchronous service invocation?' (checked); 'Required Event Parameters' with a text field for 'Event Router URL' containing 'http://localhost:7001/BEA\_MGISTICS\_1\_0\_EventRouter/'; 'Connection Pool Parameters' with fields for 'Minimum Pool Size' (1), 'Maximum Pool Size' (10), and 'Target Fraction of Maximum Pool Size' (0.7), and a checkbox for 'Allow Pool to Shrink?' (checked); 'Log Configuration' with a dropdown menu set to 'Log all messages'; and 'Configure Security' with a link 'Restrict Access to Manu using J2EE Security'. At the bottom, there are 'Deploy' and 'Save' buttons, and a checkbox for 'Deploy persistently?' (checked).

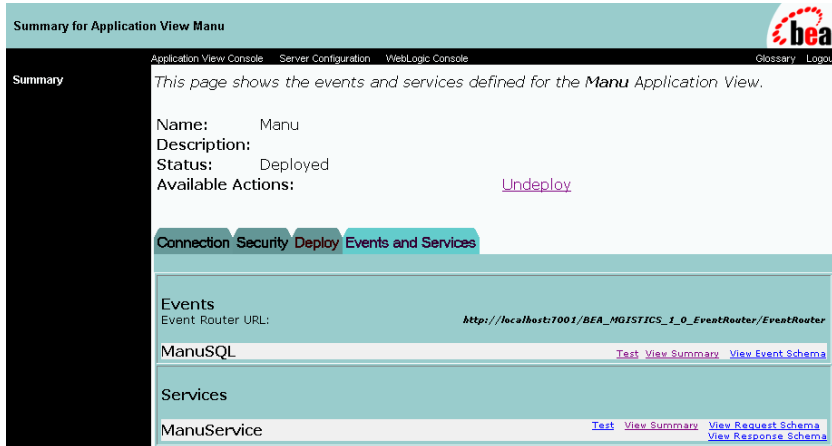
3. Update service parameters, connection pool parameters, log configuration, and security as required. For more information, see “Defining an Application View” in *Using Application Integration*:

See <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

4. Click Deploy to save and deploy the service adapter.

The Summary for Application View Manugistics Service window opens on successful deployment.

**Figure 3-8 Summary for Application View Window**



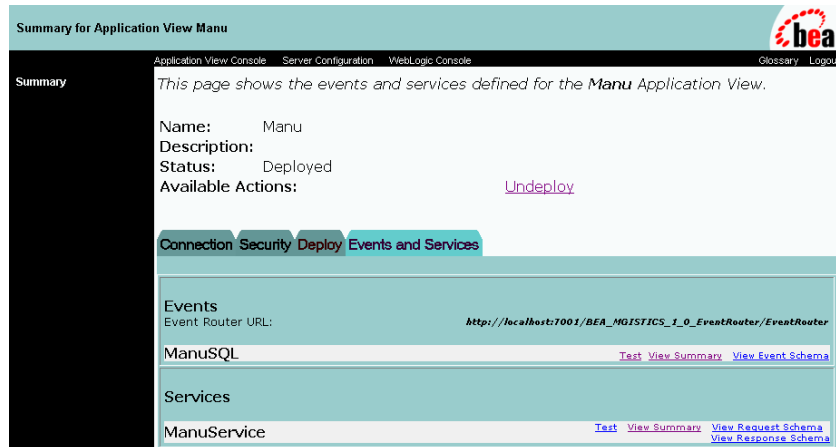
# Testing Services Using Application View Console

You can test using the Application View Console. To confirm that a deployed service adapter application view is correctly configured:

1. Log on to the WebLogic Integration Application View Console at `http://appserver-host:port/wlai`.
2. Select the folder in which the application view resides and then select the application view.

The Summary window opens.

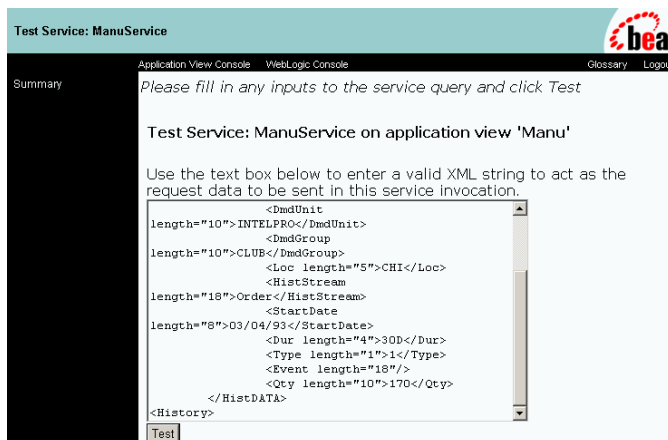
**Figure 3-9 Application View Summary**



3. Click Test.

The console displays the test entry input box.

**Figure 3-10 Test Service Window**

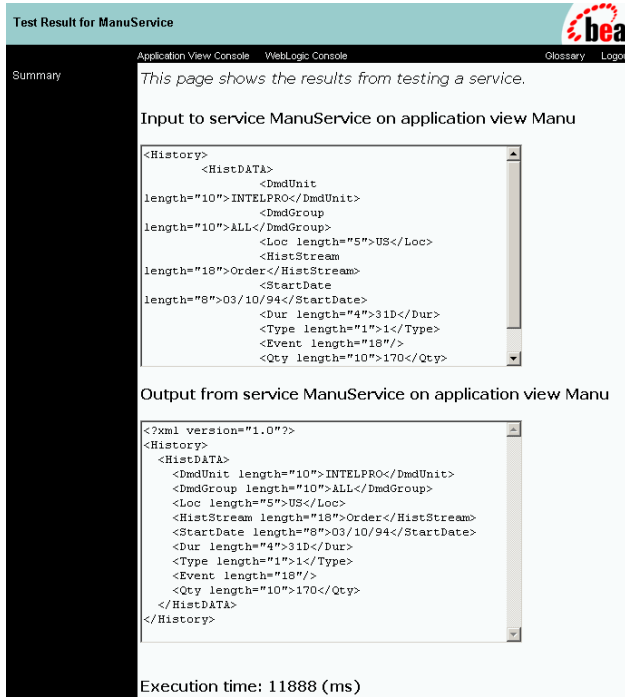


4. In the text box, enter a valid XML string that will act as the request data.

5. Click Test.

The output from the test string appears below the input box, as shown in the following figure:

**Figure 3-11 Test Service Output**



## Testing Service Adapter Application Views Using WLI Studio

You can test service adapter application views using WebLogic Integration Studio. To confirm that a deployed service adapter application view is correctly configured and that it can process services:

1. Log on to WebLogic Integration Studio:
2. In the Organization pane, select an organization to create a new workflow template.

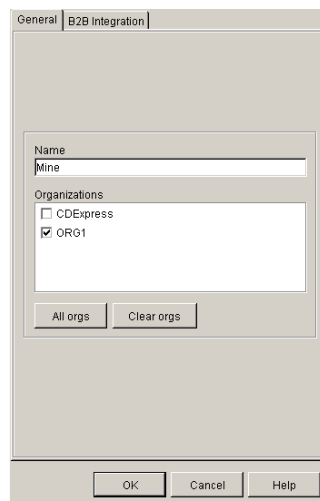
**Figure 3-12 WebLogic Integration Studio**



3. Right-click Templates and select Create Template.

The Template Properties dialog box appears.

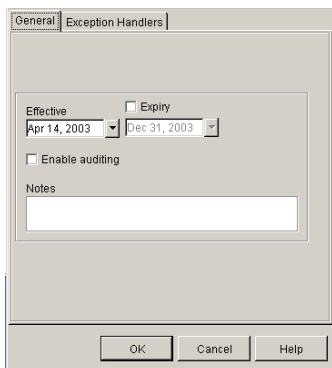
**Figure 3-13 Template Properties Dialog Box**



- a. Enter a name for your workflow template.
  - b. Click OK.
4. Right-click the new template and select Create Template Definition.  
The Template Definition dialog box opens.



**Figure 3-14 Template Definition Properties**

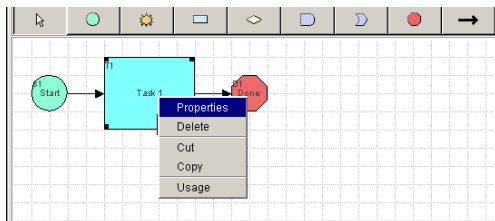


5. Click OK.

The template appears in WebLogic Integration Studio.

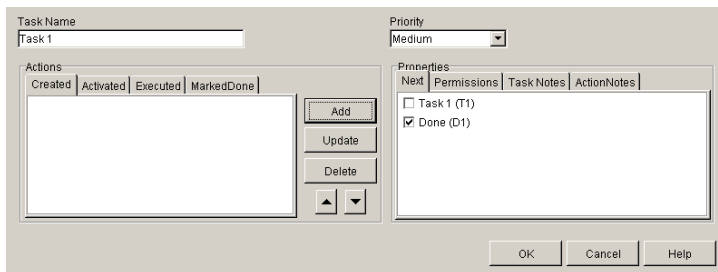
6. Right-click the Task node and select Properties.

**Figure 3-15 Choosing Node Properties in WebLogic Integration Studio**



The Task Properties dialog box opens.

**Figure 3-16 Task Properties**



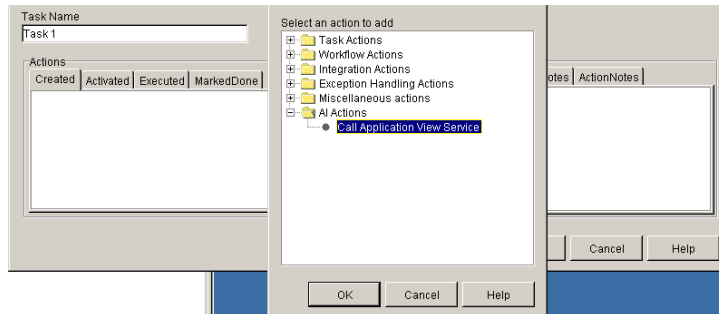
7. Select Add.

### 3 *Creating and Configuring Services*

---

The Add Action dialog box opens.

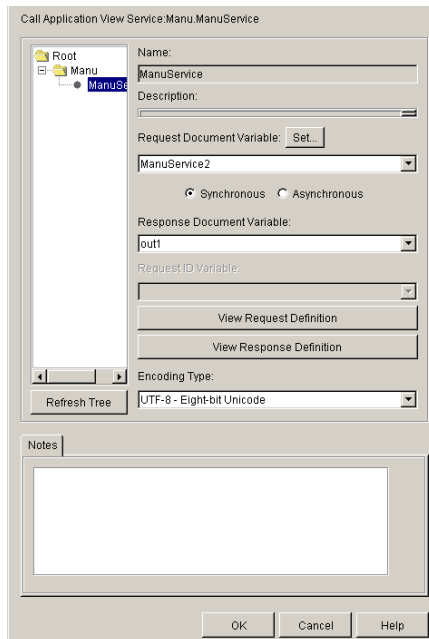
**Figure 3-17 Add Action**



8. Click AI Actions, and then Call Application View Service.

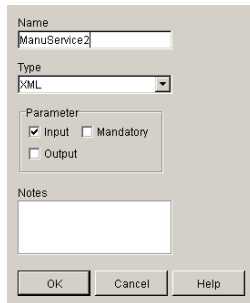
The Call Service dialog box appears.

**Figure 3-18 Call Service Dialog Box**



- a. In the service explorer in the left pane, browse the application view folders and select the application view that corresponds to the service adapter.
  - b. Open the service adapter and select the desired service.
  - c. Select <new> from the Request Document Variable drop-down list.
- The Variable Properties dialog box appears.

**Figure 3-19 Variable Properties Dialog Box**

The image shows a 'Variable Properties' dialog box. It has a 'Name' field with 'ManuService2' entered. Below it is a 'Type' dropdown menu set to 'XML'. There is a 'Parameter' section with three checkboxes: 'Input' (checked), 'Mandatory' (unchecked), and 'Output' (unchecked). At the bottom is a 'Notes' text area. At the very bottom are three buttons: 'OK', 'Cancel', and 'Help'.

- a. Enter a name for the new variable.
- b. Select the variable type XML from the drop-down list.
- c. Check the Input and Output options in the Parameter group.
- d. Click OK.

You return to the Call Service dialog box.

- a. Select Synchronous.
- b. Select New from the Response Document Variable drop-down list.
- c. Enter a name for the new variable.
- d. Select the variable type XML from the drop-down list.
- e. Click OK.
- f. Click Set in the Call Service dialog box to set the request document.

As this is only a partial workflow, the request document containing the request must be set.

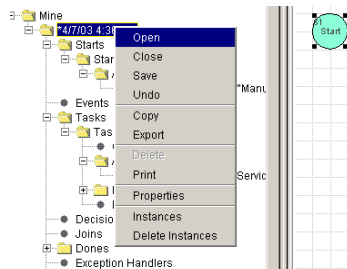
### 3 *Creating and Configuring Services*

---

This enables you to choose an XML document containing the service request.

- g. Click OK to return to the template.
9. Right-click the template in WebLogic Integration Studio's Organization pane and select Save.

**Figure 3-20 Saving a Template in WebLogic Integration Studio**



10. Right-click the new template folder you created (in this example, it is the folder below the folder named Mine) and select Properties.

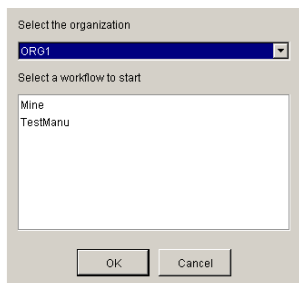
The Template Definition window opens.

- a. Ensure that Active is checked.
- b. Click OK.

You can now initiate the workflow from the BEA Worklist.

11. From the Start menu, select BEA WebLogic Platform 7.0, WebLogic Integration 7.0, and then Worklist.
12. Log on to Worklist.
13. Select Start a Workflow from the Workflow menu.

**Figure 3-21 WebLogic Integration Worklist Window - Workflow Start**



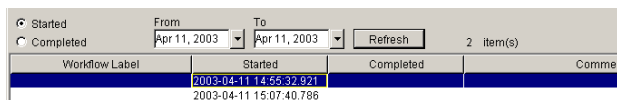
- a. Select the workflow that was just created from the workflow list.
- b. Click OK.

You receive a message indicating that your workflow started successfully.

The Workflow Instances for your service definition appear. You can now track the execution of your workflow.

14. Right-click the new template definition folder you created and select Instances.

**Figure 3-22 Choosing Instances in WebLogic Integration Studio**

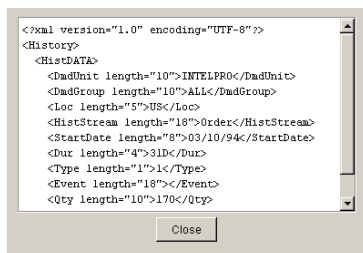


15. Right-click any instance of the workflow and select Variables.

The Workflow Variables window opens.

16. Click View XML to see the entire contents of the workflow message (also known as a document).

**Figure 3-23 View XML Window in WebLogic Integration Studio**





# 4 Creating Schema Repositories

This section addresses the schema repositories, manifests, and schemas that describe the documents entering and exiting a WebLogic Integration system. Normally, end-system metadata is used by the BAE to generate XML schemas that the adapter uses for events and services. Without this metadata, the Application Explorer can't generate these schemas for you. Because Manugistics doesn't expose any of the metadata the BEA Application Explorer uses to generate the XML schemas, you must create these schemas. After you have created the schema repository and schemas, you can configure services and events. This section includes the following topics:

- [Introduction to Schemas and Repositories](#)
- [Naming a Schema Repository](#)
- [The Repository Manifest](#)
- [Creating a Schema](#)

## Introduction to Schemas and Repositories

You describe all the documents entering and exiting your WebLogic Integration system using W3C XML schemas. These schemas describe each event arriving at and propagating from an event adapter, as well as each request sent to and each response

received from a service adapter. There is one schema for each event, and there are two for each service (one for the request and one for the response). The schemas are usually stored in files with an `.xsd` extension.

Use the Application View Console to access events and services and to assign a schema to each event, request, and response. You assign each application view to a schema repository; several application views can be assigned to the same repository.

All BEA WebLogic adapters use a schema repository to store their schema information for use by the Application View Console. The schema repository is a directory containing:

- A manifest file that describes the event and service schemas.
- The corresponding schema descriptions.

To work with schemas, you must:

- Name a schema repository.
- Create a manifest.
- Create a schema.

# Naming a Schema Repository

The schema repository has a three-part naming convention:

`session_base_directory\adapter_type\connection_name`

Here:

- `session_base_directory` is the schema's session base path, which represents a folder where schemas from multiple sessions can be held.
- `adapter_type` is the type of adapter (for example, Manugistics, MQSeries, or SAP).
- `connection_name` is a name representing a particular instance of the adapter type. For example, ManugisticsTest may be a test system, and ManugisticsProd



may be a production system; each of these systems having different events and services relevant to it.

For example, if the session base path is `/usr/opt/bea/bse`, if the adapter type is Manugistics, and if the connection name is ManugisticsProd, then the schema repository is the directory:

```
/usr/opt/bea/bse/manugistics/ManugisticsProd
```

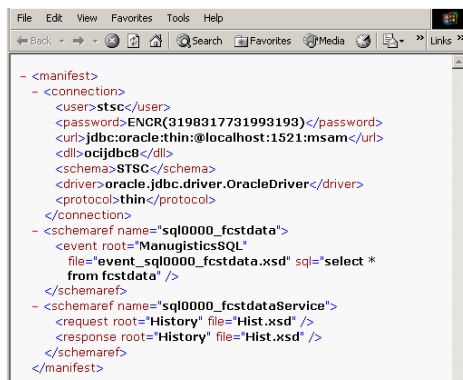
# The Repository Manifest

Each schema repository has a manifest that describes the repository and its schemas. This repository manifest is stored as an XML file named `manifest.xml`.

**Note:** If you manually create an XML schema, the namespace prefix in the manifest file must be `xsd:`.

The following is a sample manifest file:

**Figure 4-1 Displaying a Sample Manifest File in a Web Browser**



The manifest has a connection section and a schema reference section, named `schemaref`. The connection section contains default values that are used when creating the application view. The schema reference name appears in the schema

drop-down list on the Add Service and Add Event screens in the Application View Console. Each named schema reference can contain three schemas, one for each of the following schema types:

- Request specifies a request document to be sent to a service adapter.
- Response specifies a response document received from a service adapter.
- Event specifies an event that invokes an event adapter.

## Creating a Repository Manifest

The repository manifest is an XML file with the root element manifest and two sub-elements:

- `connection` appears once and is optional for the BEA WebLogic Adapter for Manugistics. If you do not want to enter default values, then include only the XML tags with no values.
- `schemaref` appears multiple times, once for each schema name. It contains all three schemas—request, response, and event.

To create a manifest:

Create an XML file with the following structure:

```
<manifest>
<connection>
  <user>stsc</user>
  <password>ENCR(3198317731993193)</password>
  <url>jdbc:oracle:thin:@localhost:1521:msam</url>
  <dll>ocijdbc8</dll>
  <schema>STSC</schema>
  <driver>oracle.jdbc.driver.OracleDriver</driver>
  <protocol>thin</protocol>
</connection>
<schemaref name="sql0000_fcstdataService">
.
.
.
</manifest>
```

1. For each new event or service schema you define, create a `schemaref` section using the following model:

```
</schemaref name="sql0000_fcstdata">
  <event root="ManugisticsSQL"
file="event_sql0000_fcstdata.xsd" sql="select * from fcstdata"/>
</schemaref>
<schemaref name="sql0000_fcstdataService">
  <request root="History" file="Hist.xsd"/>
  <response root="History" file="Hist.xsd"/>
</schemaref>
```

Here, the value you assign to:

- `file` is the name of the file in the schema repository.
- `root` is the name of the root element in the actual instance documents that arrive at, or are sent to, the event or service adapters.

## Creating a Schema

Schemas describe the rules of the XML documents that traverse WebLogic Integration. You can generate a schema manually or by using a schema-generating tool.

**Note:** If you manually create an XML schema, the namespace prefix in the manifest file must be `xsd:`.

The following listing is an example of an instance document for the `ManuSQL` event referred to in [“Creating a Repository Manifest.”](#)

### Listing 4-1 Instance Document for the `sql0000_fcstdata` Event

---

```
<row>
<DMDUNIT type="12">120-30-115</DMDUNIT>
<DMDGROUP type="12">MASSMERCH</DMDGROUP>
<LOC type="12">US</LOC>
<STARTDATE type="93">1996-09-01 00:00:00.0</STARTDATE>
<DUR type="2">43200</DUR>
<TYPE type="2">1</TYPE>
<FCSTID type="12"> </FCSTID>
<QTY type="2">2578.5205761035</QTY>
<SCEN type="2">0</SCEN>
<SCENBITS type="2">0</SCENBITS>
```

```
<MODEL type="12">DEFAULT-MODEL</MODEL>
<CURRDATE type="93">2003-04-02 16:04:59.0</CURRDATE>
</row>
```

---

A schema matching the previous instance document is shown in the following listing and can be manually coded or generated from the XML editor of your choice.

### **Listing 4-2 Schema for the sql0000\_fcstdata Instance Document**

---

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="ManugisticsSQL">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="row" minOccurs="0"
maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="DMDUNIT" minOccurs="0">
                <xsd:complexType>
                  <xsd:simpleContent>
                    <xsd:extension base="xsd:string">
                      <xsd:attribute name="type"
type="xsd:string" use="required"/>
                      <xsd:attribute name="null"
type="xsd:string" use="optional"/>
                    </xsd:extension>
                  </xsd:simpleContent>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="DMDGROUP" minOccurs="0">
                <xsd:complexType>
                  <xsd:simpleContent>
                    <xsd:extension base="xsd:string">
                      <xsd:attribute name="type"
type="xsd:string" use="required"/>
                      <xsd:attribute name="null"
type="xsd:string" use="optional"/>
                    </xsd:extension>
                  </xsd:simpleContent>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="LOC" minOccurs="0">
```

```

        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="type"
type="xsd:string" use="required"/>
                    <xsd:attribute name="null"
type="xsd:string" use="optional"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="STARTDATE" minOccurs="0">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="type"
type="xsd:string" use="required"/>
                    <xsd:attribute name="null"
type="xsd:string" use="optional"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DUR" minOccurs="0">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="type"
type="xsd:string" use="required"/>
                    <xsd:attribute name="null"
type="xsd:string" use="optional"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="TYPE" minOccurs="0">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="type"
type="xsd:string" use="required"/>
                    <xsd:attribute name="null"
type="xsd:string" use="optional"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="FCSTID" minOccurs="0">
        <xsd:complexType>

```

```

                                <xsd:simpleContent>
                                    <xsd:extension base="xsd:string">
                                        <xsd:attribute name="type"
type="xsd:string" use="required"/>
                                        <xsd:attribute name="null"
type="xsd:string" use="optional"/>
                                    </xsd:extension>
                                </xsd:simpleContent>
                            </xsd:complexType>
                        </xsd:element>
                    <xsd:element name="QTY" minOccurs="0">
                        <xsd:complexType>
                            <xsd:simpleContent>
                                <xsd:extension base="xsd:string">
                                    <xsd:attribute name="type"
type="xsd:string" use="required"/>
                                    <xsd:attribute name="null"
type="xsd:string" use="optional"/>
                                </xsd:extension>
                            </xsd:simpleContent>
                        </xsd:complexType>
                    </xsd:element>
                <xsd:element name="SCEN" minOccurs="0">
                    <xsd:complexType>
                        <xsd:simpleContent>
                            <xsd:extension base="xsd:string">
                                <xsd:attribute name="type"
type="xsd:string" use="required"/>
                                <xsd:attribute name="null"
type="xsd:string" use="optional"/>
                            </xsd:extension>
                        </xsd:simpleContent>
                    </xsd:complexType>
                </xsd:element>
            <xsd:element name="SCENBITS" minOccurs="0">
                <xsd:complexType>
                    <xsd:simpleContent>
                        <xsd:extension base="xsd:string">
                            <xsd:attribute name="type"
type="xsd:string" use="required"/>
                            <xsd:attribute name="null"
type="xsd:string" use="optional"/>
                        </xsd:extension>
                    </xsd:simpleContent>
                </xsd:complexType>
            </xsd:element>
        <xsd:element name="MODEL" minOccurs="0">
            <xsd:complexType>
                <xsd:simpleContent>
```

```

        <xsd:extension base="xsd:string">
            <xsd:attribute name="type"
type="xsd:string" use="required"/>
            <xsd:attribute name="null"
type="xsd:string" use="optional"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
</xsd:element>
<xsd:element name="CURRDATE" minOccurs="0">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="type"
type="xsd:string" use="required"/>
                <xsd:attribute name="null"
type="xsd:string" use="optional"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
    <xsd:attribute name="table" type="xsd:string"
use="optional"/>
</xsd:complexType>
</xsd:element>
</xsd:schema>

```





---

# 5 Using Tracing

Tracing is an essential feature of an adapter. Most adapters integrate different applications and do not interact with end users while processing data. Unlike a front-end component, when an adapter encounters an error or a warning condition, the adapter cannot stop processing and wait for an end user to respond.

Moreover, many business applications that are connected by adapters are mission-critical. For example, an adapter might maintain an audit report of every transaction with an EIS. Consequently, adapter components must provide both accurate logging and auditing information. The adapter tracing and logging framework is designed to accommodate both logging and auditing.

This section describes tracing for services and events. It contains the following topics:

- [Levels and Categories of Tracing](#)
- [Tracing and Performance](#)
- [Creating Traces for Services and Events](#)

# Levels and Categories of Tracing

Tracing is provided by both the BEA adapter framework and by the BEA WebLogic Adapter for Manugistics. The BEA WebLogic Integration framework provides five distinct levels of tracing:

**Table 5-1 Levels of Tracing**

Level	Indicates
AUDIT	An extremely important log message related to the business processing performed by an adapter. Messages with this priority are always written to the log.
ERROR	An error in the adapter. Error messages are internationalized and localized for the user.
WARN	A situation that is not an error, but that could cause problems in the adapter. Warning messages are internationalized and localized for the user.
INFO	An informational message that is internationalized and localized for the user.
DEBUG	A debug message, that is, information used to determine how the internal elements of a component are working. Debug messages usually are not internationalized.

The adapter framework provides three specialized categories of tracing:

**Table 5-2 Trace Categories**

Level	Indicates
Basic Trace	Basic traces. Displays the input XML (up to 300 bytes) before parsing, and shows the request being processed. The default setting is off.
Verbose Trace	More extensive traces. Displays configuration parameters used by the adapter. The default setting is off.

**Table 5-2 Trace Categories**

Level	Indicates
Document Trace	Displays the input document after it was analyzed and the response document being returned. Because some documents are very large, this trace category can severely affect performance and memory use. The default setting is off.

**Note:** To obtain the appropriate trace, both the level and the category must be declared. In a debug situation, BEA Customer Support will request (minimally) a Basic and a Verbose trace.

## Tracing and Performance

The additional trace capabilities provided by the adapter are not strictly hierarchic; rather they are categorized. These traces are designed to provide debugging help with minimum effect on performance. All internal adapter traces are controlled through the additional tracing settings, and all additional settings route their output to the standard debug setting.

If you configure the adapter for additional settings and do not configure standard trace settings, the traces are generated but never appear in output. This affects performance, as the production of the trace continues even though you receive no benefit of the additional trace information.

## Creating Traces for Services and Events

The following topics discuss the steps required to create traces to diagnose adapter problems.

## Creating Traces for a Service

To create traces for a service:

1. Create or modify the service.
2. Ensure that all of the adapter parameters are entered correctly.
3. Select the appropriate schema from the drop-down list.
4. Select the appropriate trace levels as described in [Table 5-2](#) Trace, Verbose trace, and Document trace.

**Figure 5-1 Add Service window**

**Add Service**

Application View Console WebLogic Console

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

Unique Service Name: \*

**Manugistics**

Drive	~Drive~
ManubaPath*	~ManubaPath~
BatchPath*	~BatchPath~
LSTFile*	~LSTFile~
LOGFile*	~LOGFile~
OutputFile*	~OutputFile~
encoding*	UTF-8

schema: sql0000\_fcstdataService

**settings**

Trace on/off	<input type="checkbox"/>
Verbose Trace on/off	<input type="checkbox"/>
Document Trace on/off	<input type="checkbox"/>

Add

5. Click Add to continue to the next configuration pane.
  6. Click Continue to move to the next configuration pane.
- The Deploy Application View window opens.
7. Navigate to the Log Configuration area and select the desired trace level.
- This pane enables you to select the trace level for the BEA WebLogic Integration framework.

**Figure 5-2 Deploy Application View window**

**Deploy Application View Manu to Server**

Application View Console | WebLogic Console | Glossary | Logout

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

**Required Service Parameters**

Enable asynchronous service invocation? ☒

**Required Event Parameters**

Event Router URL \*

**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.

Log all messages  
Log errors and audit messages  
Log warnings, errors, and audit messages  
Log informationals, warnings, errors, and audit messages  
Log all messages

For maximum tracing, select Log all Messages.

This is recommended to obtain optimum debugging information for BEA support personnel.

**Note:** This causes all generated messages to be written to the log. You must select the desired category as defined in [Table 5-2](#) in the adapter to generate the required messages.

8. Click Deploy (or Save) to set the trace settings and deploy the application view.

Traces are created the next time the service is invoked.

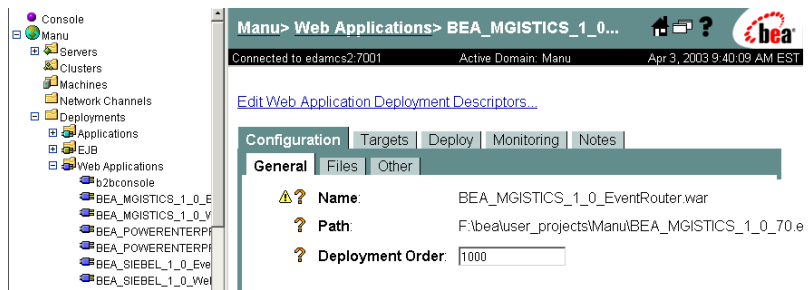
Traces are output to a file named BEA\_MGISTICS\_1\_0\_1.log in the WebLogic Domain home directory.

## Creating or Modifying the Tracing Level for an Event

To create or modify the WebLogic framework tracing level for an event:

1. Logon to the BEA WebLogic Server Console.
2. In the left pane, select Web Applications.
3. Select `BEA_MGISTICS_1_0_EventRouter.war`.

**Figure 5-3 WebLogic Server Console EventRouter Configuration**



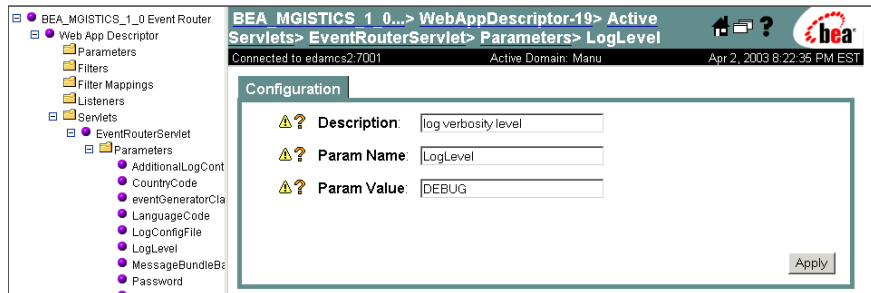
4. Click Edit Web Application Deployment Descriptors.

A new window opens.

5. In the left pane, select Servlets.
6. In the folder below Servlets, select EventRouterServlet.
7. Select Parameters.
8. Select LogLevel.

The log level configuration parameters appear in the right pane.

**Figure 5-4 WebLogic Server Console: Configuration**



This pane enables you to select the trace level for the BEA WebLogic Integration framework.

For maximum tracing, enter DEBUG. This is recommended to obtain optimum debugging information for BEA support personnel.

The following levels are valid:

**Table 5-3 Tracing Levels**

Level	Indicates
AUDIT	An extremely important log message related to the business processing performed by an adapter. Messages with this priority are always written to the log.
ERROR	An error in the adapter. Error messages are internationalized and localized for the user.
WARN	A situation that is not an error, but that could cause problems in the adapter. Warning messages are internationalized and localized for the user.
INFO	An informational message that is internationalized and localized for the user.
DEBUG	A debug message, that is, information used to determine how the internals of a component are working. Debug messages usually are not internationalized.

9. Click Apply to save the newly entered trace level.

10. Click the BEA\_MGISTICS\_1\_0 EventRouter.

11. Click Persist to apply the logging changes.

This change need only be made once. It is set for all events associated with a given adapter.

12. Return to the WebLogic Server Console.
13. Select Applications from the WebLogic Server Console.
14. Select the adapter whose EventRouter you have modified in the previous steps.
15. Select the Deploy tab in the right pane.

The right pane displays the following adapter components:

- BEA\_MGISTICS\_1\_0.rar
- BEA\_MGISTICS\_1\_0.web.rar
- BEA\_MGISTICS\_1\_0\_EventRouter.war

**Figure 5-5 WebLogic Server Console: Redeploy**

Manu> Applications> BEA\_MGISTICS\_1\_0

Connected to edamcs27001 Active Domain: Manu Apr 3, 2003 9:58:51 AM EST

[Edit Application Descriptor...](#)

Configuration **Deploy** Notes

**Deployment Status by Target:**

Component	Component Type	Target	Target Type	Deployed	
<a href="#">BEA_MGISTICS_1_0_EventRouter.war</a>	Web Application	myserver	Server	true	<input type="button" value="Undeploy"/> <input type="button" value="Redeploy"/>
<a href="#">BEA_MGISTICS_1_0.rar</a>	Connector Component	myserver	Server	true	<input type="button" value="Undeploy"/> <input type="button" value="Redeploy"/>
<a href="#">BEA_MGISTICS_1_0_Web.war</a>	Web Application	myserver	Server	true	<input type="button" value="Undeploy"/> <input type="button" value="Redeploy"/>

Undeploy the entire application

16. Redeploy the EventRouter by clicking the Redeploy button to the right of BEA\_MGISTICS\_1\_0\_EventRouter.war.



# Creating Adapter Logs for an Event

To create adapter logs for an event:

1. Create or modify the event.
2. Ensure that all of the adapter parameters are entered correctly.

**Figure 5-6 Add Event window**

The screenshot shows the 'Add Event' window in the BEA WebLogic console. The left sidebar contains a navigation menu with options: 'Configure Connection', 'Administration', 'Add Service', 'Add Event' (highlighted), and 'Deploy Application View'. The main content area has a header with 'Application View Console' and 'WebLogic Console' tabs, and a 'Glossary' link. Below the header, a message states: 'On this page, you add events to your application view.' The 'Unique Event Name:' field is empty. The 'ManugisticsSQL' adapter is selected, showing a form with the following fields: 'Character Set Encoding\*' (UTF-8), 'Driver\*' (oracle.jdbc.driver.OracleDriver), 'Url\*' (jdbc:oracle:thin:@localhost:1521:ms), 'User Name' (stsc), 'Password' (masked with asterisks), 'Format\*' (field dropdown), 'Maximum Rows' (5000), 'SQL Post Query' (empty), 'Delete Keys' (empty), 'Polling Interval' (180), and 'Data Source Name' (empty). Below the form, the 'schema:' dropdown is set to 'sq0000\_fctdata'. The 'settings' section contains three checkboxes: 'Trace on/off', 'Verbose Trace on/off', and 'Document Trace on/off', all of which are unchecked. An 'Add' button is located at the bottom of the form.

3. Select the appropriate schema from the drop-down list.
4. Select the appropriate trace levels as described in [Table 5-2](#): Trace, Verbose trace, and Document trace.
5. Click Add to continue to the next configuration pane.
6. Click Continue to move to the next configuration pane.

The Deploy Application View window opens.

7. Navigate to the Log Configuration area and select the desired trace level.

This pane enables you to select the trace level for the BEA WebLogic Integration framework.

**Figure 5-7 Deploy Application View window**

For maximum tracing, select Log all Messages. This is recommended to obtain optimum debugging information for BEA support personnel.

8. Click Deploy (or Save) to set the trace settings and deploy the application view.

Traces are created the next time the event occurs.

Traces are output to a file named `BEA_MGISTICS_1_0.log` in the WebLogic Domain home directory.