



BEA WebLogic Adapter for MQSeries®

User Guide

Copyright

Copyright © 2002 BEA Systems, Inc. All Rights Reserved.

Copyright © 2002 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 MQSeries User Guide

Part Number	Date
N/A	October 2002

Table of Contents

About This Document

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

1. Introducing the BEA WebLogic Adapter for MQSeries

Introduction	1-1
How the BEA WebLogic Adapter for MQSeries Works	1-2

2. Metadata, Schemas, and Repositories

Understanding Metadata.....	2-2
Schemas and Repositories	2-5
Naming a Schema Repository	2-6
The Repository Manifest	2-7
Creating a Repository Manifest.....	2-8
Creating a Schema	2-9
Storing Directory and Template Files for Transformations	2-12
Samples File	2-12

3. Creating and Configuring an Event Adapter

Creating an Application View Folder.....	3-2
Creating an Event Adapter Application View	3-4
Configuring an Event Adapter Application View	3-10
Testing Event Adapter Application Views Using Application View Console	3-17
Testing Event Adapter Application Views Using WebLogic Integration Studio	3-24

4. Creating and Configuring a Service Adapter

Creating a Service Adapter Application View	4-1
Configuring a Service Adapter Application View.....	4-5
Testing the BEA Service Adapter for MQSeries.....	4-14

5. Transforming Document Formats

Kinds of Transformations	5-1
Non-XML to XML Transformation	5-2
XML to XML Transformations.....	5-6
XML to Non-XML Transformations.....	5-8
Creating MFL Transformations	5-11
Testing MFL Transformations.....	5-13

A. Error Messages and Troubleshooting

About This Document

This document explains how to configure and use the BEA WebLogic Adapter for MQSeries to integrate IBM MQSeries messages with BEA WebLogic Integration.

This document covers the following topics:

- [Chapter 1, “Introducing the BEA WebLogic Adapter for MQSeries,”](#) provides an overview of the adapter’s benefits and processes.
- [Chapter 2, “Metadata, Schemas, and Repositories,”](#) describes how metadata for your enterprise information system (EIS) is described, how to name a schema repository and the schema manifest, how to create a schema, and how to store directory and template files for transformations.
- [Chapter 3, “Creating and Configuring an Event Adapter,”](#) describes how to create, configure, and test an event adapter.
- [Chapter 4, “Creating and Configuring a Service Adapter,”](#) describes how to create, configure, and test a service adapter.
- [Chapter 5, “Transforming Document Formats,”](#) describes how you can convert documents between XML and non-XML formats using several transformation phases.
- [Appendix A, “Error Messages and Troubleshooting,”](#) lists error messages and suggests what action to take for each message if displayed.

What You Need to Know

This document is written for system integrators who develop client interfaces between MQSeries and other applications. It describes how to use the BEA WebLogic Adapter for MQSeries and how to develop application environments with specific focus on message integration. It is assumed that readers know Web technologies and have a general understanding of Microsoft Windows and UNIX systems.

Related Information

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

- *BEA WebLogic Adapter for MQSeries Installation and Configuration Guide*
- *BEA WebLogic Adapter for MQSeries Release Notes*
- *BEA Application Explorer Installation Guide*
- 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 MQSeries 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 BEA WebLogic Adapter for MQSeries documentation.

In your e-mail message, please indicate which version of the BEA WebLogic Adapter for MQSeries documentation you are using.

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

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

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

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.
<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> <pre>#include <iostream.h> void main () the pointer psz chmod u+w * \tux\data\ap .doc tux.doc BITMAP float</pre>
monospace boldface text	Identifies significant words in code. <i>Example:</i> <pre>void commit ()</pre>
<i>monospace italic text</i>	Identifies variables in code. <i>Example:</i> <pre>String <i>expr</i></pre>
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. <i>Examples:</i> <pre>LPT1 SIGNON OR</pre>

Convention	Item
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
[]	Indicates optional items in a syntax line. The brackets themselves should never be typed. <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">■ That an argument can be repeated several times in a command line■ That the statement omits additional optional arguments■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. <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 MQSeries

This section provides a summary of the key features of the BEA WebLogic Adapter for MQSeries and describes how it works. It includes the following topics:

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

Introduction

The BEA WebLogic Adapter for MQSeries integrates your IBM MQSeries® messages with WebLogic Integration in a fast, easy, and reliable way. You can use the adapter to exchange XML, non-XML, ASCII, and custom data formats between your MQSeries resources and WebLogic Integration to provide a tightly integrated and reliable application infrastructure. (Note that IBM has renamed MQSeries “WebSphere MQ.”)

MQSeries messaging products support application integration by sending and receiving data as messages that allow business applications to exchange information across different platforms. They account for network interfaces, assure “once only” delivery of messages, deal with communication protocols, dynamically distribute workload across available resources, handle recovery after system problems, and help make programs portable. This allows programmers to use their skills to handle key business requirements, instead of wrestling with underlying network complexities.

The BEA WebLogic Adapter for MQSeries provides:

- *Guaranteed asynchronous, bi-directional message interactions* between WebLogic Integration and native IBM MQSeries managed queues.
- *Data transfer* between a business process running within WebLogic Integration and an MQSeries Queue Manager.
- *Service and event adapter integration operations* providing end-to-end business process management using XML schemas.
- *Support for many formats* including XML, Comma Separated Variable (CSV), Excel, Message Format Language (MFL), and Custom Data Formats (CDF). The adapter converts non-XML files into XML formats. Delimited, fixed length, and variable length file formats are supported.

How the BEA WebLogic Adapter for MQSeries Works

The BEA WebLogic Adapter for MQSeries provides transport protocol support so that it can listen for and emit documents from MQSeries queues using the MQSeries resource manager, called the Queue Manager. Transaction integrity is maintained at all times. The adapter can accept messages arriving on a named queue and can route these messages to any queue or any other adapter.

The listening capability has been implemented as an event adapter within WebLogic Integration. When an inbound document is detected, the event adapter provides a number of options that you can configure with the design-time WebLogic Application View Console:

- **Transformation services.** XML is quickly becoming the standard for exchanging information between applications; it is invaluable in integrating disparate applications. With this in mind, and recognizing that the world does not yet speak XML exclusively, the BEA WebLogic Adapter for MQSeries provides transformation services.

The adapter uses pre-built customizable parsers to enable the parsing and conversion of non-XML formatted documents, and XSLT transformation to modify XML document formats, ensuring that any incoming document can be converted to XML as specified by your event and service schemas. You can also use the adapter in conjunction with other BEA WebLogic Adapters to handle the processing of various types of message types, such as SAP IDoc, SWIFT, FIX, HIPAA, and HL7.

- **Document validation rules.** During the analysis of an incoming document, you can invoke one or more user exits to examine and transform parts of the document.

For example, you can create a name and address validation exit that gets called each time such a set of elements appears in the document. The same exit logic might apply to all documents, so that all names and addresses in a complete system are validated in the same manner. Validation rule specifications are stored in XML files that are freely accessible in the directory structure. Keeping each rule in an external file facilitates the maintenance of existing rules, and provides an easy way to add new ones. You can also create new rules by writing custom Java code.

- **Protocol emitting.** The service adapter supports emitting documents to an MQSeries queue.
- **Document chaining.** You can use any service adapter to enhance the functionality of a document flow. You can chain any number of desired service adapters together.

For example, you can chain multiple protocol agents to send a message to multiple transports and locations.

2 Metadata, Schemas, and Repositories

This section explains how metadata for your enterprise information system (EIS) is described, how to name a schema repository and the schema manifest, how to create a schema, and how to store directory and template files for transformations. After the metadata for your EIS is described, you can create and deploy application views using the WebLogic Application View Console.

This section includes the following topics:

- [Understanding Metadata](#)
- [Schemas and Repositories](#)
- [The Repository Manifest](#)
- [Creating a Schema](#)
- [Storing Directory and Template Files for Transformations](#)

Understanding Metadata

When you define an application view, you are creating an XML-based interface between WebLogic Integration and an enterprise information system (EIS) or application within your enterprise. The BEA WebLogic Adapter for MQSeries is used to define a file based interface to applications within and outside of the enterprise. Many applications or information systems use file systems to store and share data. These files contain information required by other applications, and this information can be fed information via the BEA WebLogic Adapter for MQSeries.

The BEA WebLogic Adapter for MQSeries can read, write, or manipulate different types of files stored in multiple file systems or FTP sites. WebLogic integration uses XML as the common format for data being processed in its workflows, which requires information that is not in XML to be transformed to XML. Alternatively, to share information successfully, the adapter can transform information from the XML format used in WebLogic Integration to widely used formats, such as commercial XML schemas, EDI, SWIFT, HIPAA, HL7, and others.

For example, Excel is a widely used application that allows all types of professionals (from fund managers to administrative assistants) to collate information pertinent to their working environment. This information can be shared by other applications using the adapter's transformation capability, which can convert a worksheet to XML and to other business partners via an EDI stream.

To map this information within the workflow via event and service adapters, the BEA WebLogic Adapter for MQSeries requires XML schemas for identifying and processing these documents. Because some of these documents may be in non-XML form, such as Excel, CSV, SWIFT, or HIPAA, they must be converted to XML and described to WebLogic Integration using these schemas. A manifest file is used to relate schemas to events or services. The schemas and manifest are stored in a folder or directory in the local file system referred to as the EIS repository. The repository location is required when creating an application view from which events and services can be configured.

Events are triggers to workflows. When a particular file arrives at a location, an event can be triggered to read and convert, if necessary, to the XML format that conforms to a particular schema, which then initiates a flow. Services are called from the workflow to perform supported operations.

The adapter converts non-XML, non-self describing documents into XML in two ways. The Format Builder tool can build MFL files that are stored in the WebLogic server local repository. The Format Builder is best used for unconventional or custom format files. The structure of this file can be defined using the Format Builder and used for basic conversion to or from XML. For conventional documents that are not self-describing, such as SWIFT, HIPAA, EDI/X12, EDIFACT, and HL7, the structure of the data is described using a data dictionary or .dic file.

Pre-built dictionaries are supplied for these formats, so creating them is not necessary, but you can customize them to conform with specific electronic trading agreements. Transformation templates or .xch files use these dictionaries to map the document to its XML form or vice versa.

Transformation templates use dictionaries as metadata for the file being read or created. The template defines the input value's relationship with the output values using the dictionary and XML schema. For events, the template is used to convert a non-XML format to XML and for services, the conversion can be reversed using an alternative template.

The templates are stored in the templates sub-directory of the EIS repository. Dictionaries are stored in the dictionaries sub-directory. The following is a sample data dictionary.

Listing 2-1 Data Dictionary Sample

```
<?xml version="1.0"?>
<!-- Title = EDI Transaction Dictionary by Transaction Set -->
<!-- Transaction = 276 Health Care Claim Status Request -->

<EDI Type="ASCII" Version="4010" Standard="X12">
<TransactionSet ID="276" Name="Health Care Claim Status Request"
Note="">

<!-- Table 1 -->

    <Segment ID="ST" Name="Transaction Set Header" Req="M"
MaxUse="1">

        <Element ID="01" Name="Transaction Set Identifier Code"
Req="M" Type="ID" MinLength="3" MaxLength="3" Note="The transaction
set identifier 'ST01' is used by the translation routines of the
interchange partners to select the appropriate transaction set
definition 'e.g., 810 select the Invoice Transaction Set'."/>
```

2 *Metadata, Schemas, and Repositories*

```
<Element ID="02" Name="Transaction Set Control Number" Req="M"
Type="AN" MinLength="4" MaxLength="9"/>

<Element ID="03" Name="Implementation Convention Reference"
Req="O" Type="AN" MinLength="1" MaxLength="35" Note="The
implementation convention reference 'ST03' is used by the
translation routines of the interchange partners to select the
appropriate implementation convention to match the transaction set
definition."/>

</Segment>

<Segment ID="BHT" Name="Beginning of Hierarchical Transaction"
Req="M" MaxUse="1">

<Element ID="01" Name="Hierarchical Structure Code" Req="M"
Type="ID" MinLength="4" MaxLength="4"/>

<Element ID="02" Name="Transaction Set Purpose Code" Req="M"
Type="ID" MinLength="2" MaxLength="2"/>

<Element ID="03" Name="Reference Identification" Req="O"
Type="AN" MinLength="1" MaxLength="50" Note="BHT03 is the number
assigned by the originator to identify the transaction within the
originator's business application system."/>
```

After the metadata for your EIS has been described, application views can be created and deployed using the WebLogic Integration Application View Console. For more information on creating application views, see [Chapter 3, “Creating and Configuring an Event Adapter,”](#) and [Chapter 4, “Creating and Configuring a Service Adapter.”](#)

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

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

BEA WebLogic Adapters all make use of a schema repository to store their schema information and present it to the WebLogic 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 know how to:

- 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\connection_name

- *session_base_directory* is the schema's session base path, which represents a folder under which multiple sessions of schemas may be held.
- *adapter* is the type of adapter (for example, MQSeries or SAP).
- *connection_name* is a name representing a particular instance of the adapter type. For example, QMBEA may be a connection to a particular MQSeries Queue Manager, while QMDEV may be another, with each of these systems having different events and services relevant to them.

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

`/usr/opt/bea/bse/mqseries/QMPROD`

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`.

The following is an example of a sample manifest file showing relationships between events and services and their related schemas.

The manifest file relates documents (through their schemas) to services and events. The manifest exposes schema references to the event relating the required document (via the root tag) to the corresponding schema. Schemas and manifests are stored in the same directory, the repository root of the EIS. The following is an example of the a manifest file with a description of the elements.

Listing 2-2 Sample Manifest File

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<manifest>
  <connection/>
  <schemaref name="service_only">
    <request root="INVOICE" file="INVOICE.xsd"/>
    <response root="emitStatus" file="FileEmit.xsd"/>
  </schemaref>
  <schemaref name="event_only">
    <event root="PURCHASE_ORDER" file="PURCHASE_ORDER.xsd"/>
  </schemaref>
  <schemaref name="shared">
    <request root="STOCK_STATUS" file="STOCK_STATUS.xsd"/>
    <response root="emitStatus" file="FileEmit.xsd"/>
    <event root="STOCK_UPDATE" file="STOCK_UPDATE.xsd"/>
  </schemaref>
</manifest>
```

The manifest has a connection section (which is not used by the BEA WebLogic Adapter for MQSeries) and a schema reference section, named *schemaref*. The schema reference name is displayed in the schema drop-down list on the Add Service and Add Event windows in the WebLogic Integration Application View Console. This sample manifest has three schema references or *schemaref* tags; one for services only, one for events only, and one for a combination of services and events. Events require only one schema, defined by the *event* tag. This relates the root tag of an XML document to a schema in the EIS repository. For services, two schemas are required: one for the document being passed to the service, represented by the *request* tag, and one for the expected *response* document received from the service operation, represented by the *response* tag.

Creating a Repository Manifest

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

- *connection*, which appears once, and which you can ignore because it is not used by the BEA WebLogic Adapter for MQSeries.
- *schemaref*, which appears multiple times, once for each schema name, and which contains all three schemas—request, response, and event.

To create a manifest:

1. Create an XML file with the following structure:

```
<manifest>
  <connection>
  </connection>
</manifest>
```

2. For each new event or service schema you define, create a `schemaref` section using this model:

```
<schemaref name="OrderIn">
  <request root="OrderIn" file="service_OrderIn_request.xsd"/>
  <response root="emitStatus" file="MQEmitStatus.xsd"/>
  <event root="OrderIn" file="event_OrderIn.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 will arrive at, or be sent to, the event or service.

Creating a Schema

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

WebLogic Integration interacts with application view events and services by sending and receiving XML messages. The XML messages are defined by XML schemas. The schemas are stored in directories specific for each adapter.

You must set up at least one directory for each adapter you use. This directory can contain multiple subdirectories, each of which can hold schemas specific to different instances of your application. You should name the parent directory to represent your adapter; you can name the subdirectories according to what is appropriate for your application.

For example, if you have four instances of an application that exchanges messages between the BEA WebLogic Adapter for MQSeries and WebLogic Integration, you should set up four subdirectories to store the schemas; the subdirectories should be in a parent `MQSeries` directory:

```
D: \TraderSystems\BEAapps\MQSeries\FTPprod
D: \TraderSystems\BEAapps\MQSeries\FTPdev
D: \TraderSystems\BEAapps\MQSeries\FTPuat
```

The schemas for the documents being processed are stored within those directories.

The following is an example of an instance document for the OrderIn event referred to in [“Creating a Repository Manifest” on page 2-8](#).

Listing 2-3 Instance Document for OrderIn Event

```
<?xml version="1.0"?>

<OrderIn>
  <Store_Code>1003CA</Store_Code>
  <LineItem>
    <Prod_Num>1003</Prod_Num>
    <Quantity>100</Quantity>
    <Price>1.69</Price>
  </LineItem>
  <LineItem>
    <Prod_Num>1004</Prod_Num>
    <Quantity>10</Quantity>
    <Price>1.79</Price>
  </LineItem>
</OrderIn>
```

The following is a schema matching this instance document and may be manually coded or generated from any XML editor:

Listing 2-4 Schema Matching OrderIn Event Instance Document

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xsd:element name="OrderIn">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Store_Code"/>
        <xsd:element ref="LineItem" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="LineItem">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Prod_Num"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```



```

        <xsd:element ref="Quantity"/>
        <xsd:element ref="Price"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="Price">
    <xsd:simpleType>
        <xsd:restriction base="xsd:decimal">
            <xsd:enumeration value="1.69"/>
            <xsd:enumeration value="1.79"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="Prod_Num">
    <xsd:simpleType>
        <xsd:restriction base="xsd:short">
            <xsd:enumeration value="1003"/>
            <xsd:enumeration value="1004"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="Quantity">
    <xsd:simpleType>
        <xsd:restriction base="xsd:byte">
            <xsd:enumeration value="10"/>
            <xsd:enumeration value="100"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="Store_Code" type="xsd:hexBinary"/>
</xsd:schema>

```

Storing Directory and Template Files for Transformations

The BEA WebLogic Adapter for MQSeries supports the exchange of XML and non-XML messages with WebLogic Integration. Templates and dictionaries are created and associated with the BEA WebLogic Adapter for MQSeries events and services. Dictionaries (.dic extension) are documents that describe an incoming non-XML document. Templates (.xch extension) describe the conversion from one format to another (XML to non-XML, and vice versa). Sample dictionaries and templates are supplied with the product and must be placed in a `transform` subdirectory in the root directory for your domain, as shown in the following paths:

```
DOMAIN_HOME\transform\xch  
DOMAIN_HOME\transform\xslt  
DOMAIN_HOME\transform\dic
```

Samples File

Supplied with the BEA WebLogic Adapter for MQSeries are sample files (xml and edi format) that can be used to help test that your environment is correctly set up and working. The `samples.zip` file also includes sample manifest and schema files.

3 Creating and Configuring an Event Adapter

This section describes how to create, configure, and test an event adapter application view. An event adapter is the inbound interface from an MQSeries resource manager (that is, from a Queue Manager and its associated queues) to a workflow. This section includes the following topics:

- [Creating an Application View Folder](#)
- [Creating an Event Adapter Application View](#)
- [Configuring an Event Adapter Application View](#)
- [Testing Event Adapter Application Views Using Application View Console](#)
- [Testing Event Adapter Application Views Using WebLogic Integration 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; if you wish, 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 `//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.

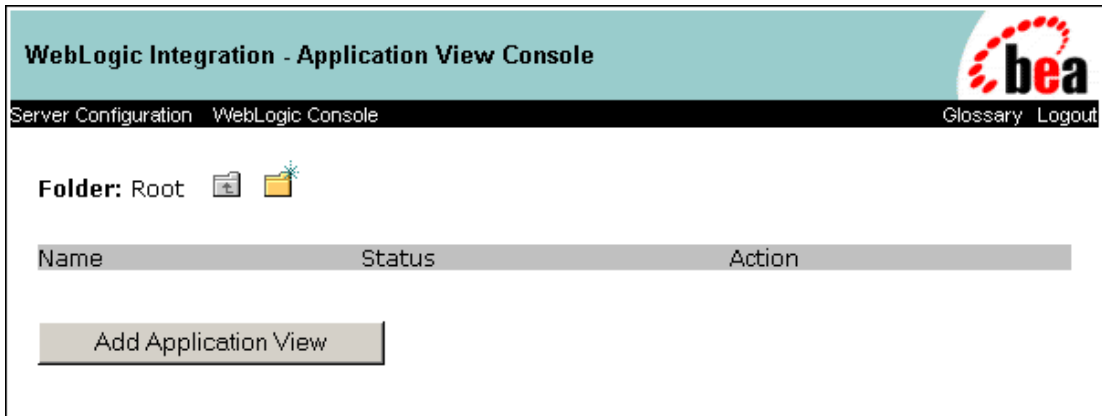
2. If prompted, enter a user name and password.

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 MQSeries Installation and Configuration Guide*.

3. Click Login.

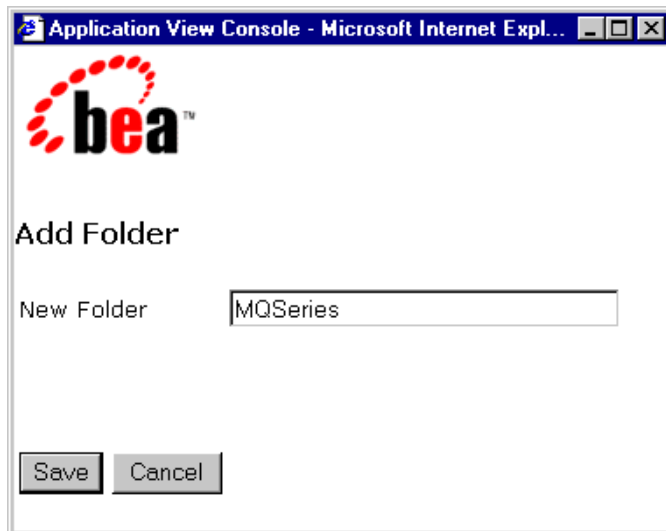
The WebLogic Integration Application View Console opens.

Figure 3-1 Application View Console Main Window



4. Double-click the new folder icon. The Add Folder window opens.

Figure 3-2 Add Folder Window



5. Supply a name for the folder, and then click Save.

You have finished creating the application view folder. To create an event adapter application view, see [“Creating an Event Adapter Application View.”](#) To create a service adapter application view, see [“Creating a Service Adapter Application View” on page 4-1.](#)

Creating an Event Adapter Application View

To create an event adapter application view:

1. Log on to the WebLogic Application View Console at
`//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.

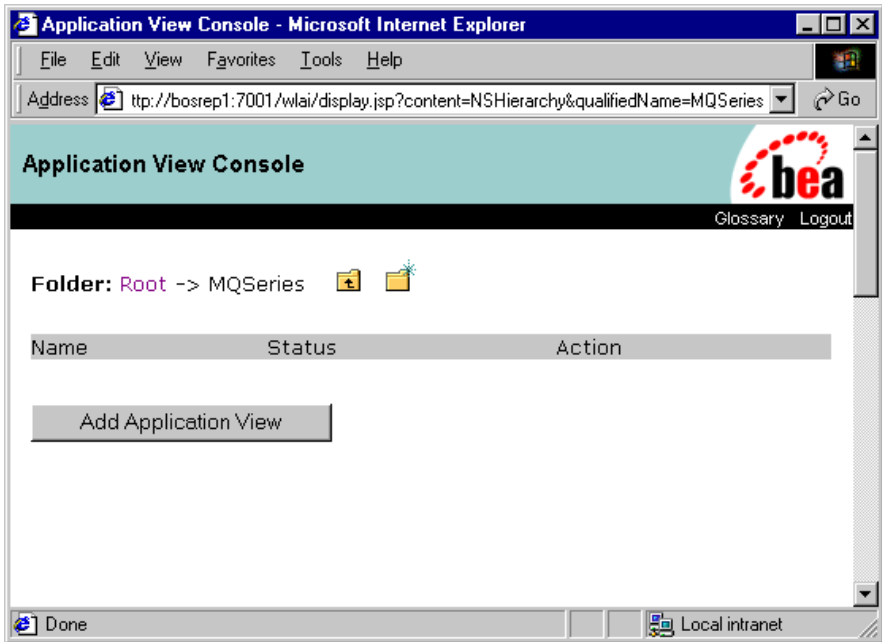
2. If prompted, enter a user name and password.

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 MQSeries Installation and Configuration Guide*.

3. Click Login.

The WebLogic Integration Application View Console opens.

Figure 3-3 Selecting the MQSeries Folder in the Main Window




4. Select the desired application view folder.
5. Click Add Application View. The Define New Application View window opens.

Figure 3-4 Define New Application View Window

Application View Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address <http://bosrep1:7001/wlai/display.jsp?content=defappvw&namespace=MQ> Go

Define New Application View  [Glossary](#) [Logout](#)

This page allows you to define a new application view

Folder: MQSeries

Application View Name: * MQEvent

Description: BEA Adapter for MQSeries - Event

Associated Adapter: BEA_MQSERIES_1_0

OK Cancel

Done Local intranet

6. Enter a name and description for the application view.
7. Select BEA_MQSERIES_1_0 from the Associated Adapter drop-down list.
8. Click OK. The Configure Connection Parameters window opens.

Figure 3-5 Configure Connection Parameters Window

Configure Connection Parameters

Application View Console WebLogic Console Glossary Logout

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

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*

Done Local intranet

3 Creating and Configuring an Event Adapter

9. Enter the name of the BEA WebLogic Adapter for MQSeries session path (sometimes known as the session base directory).

This path holds your MQSeries schema and connection information:

drive:\session_path\mqseries\connection_name\schema_files

Here,

drive is the disk drive (for example, C).

session_path, which you are entering in this window, is the absolute path to the directory you wish to contain the directory structure for your XML schema files (for example, \BEA\Schemas).

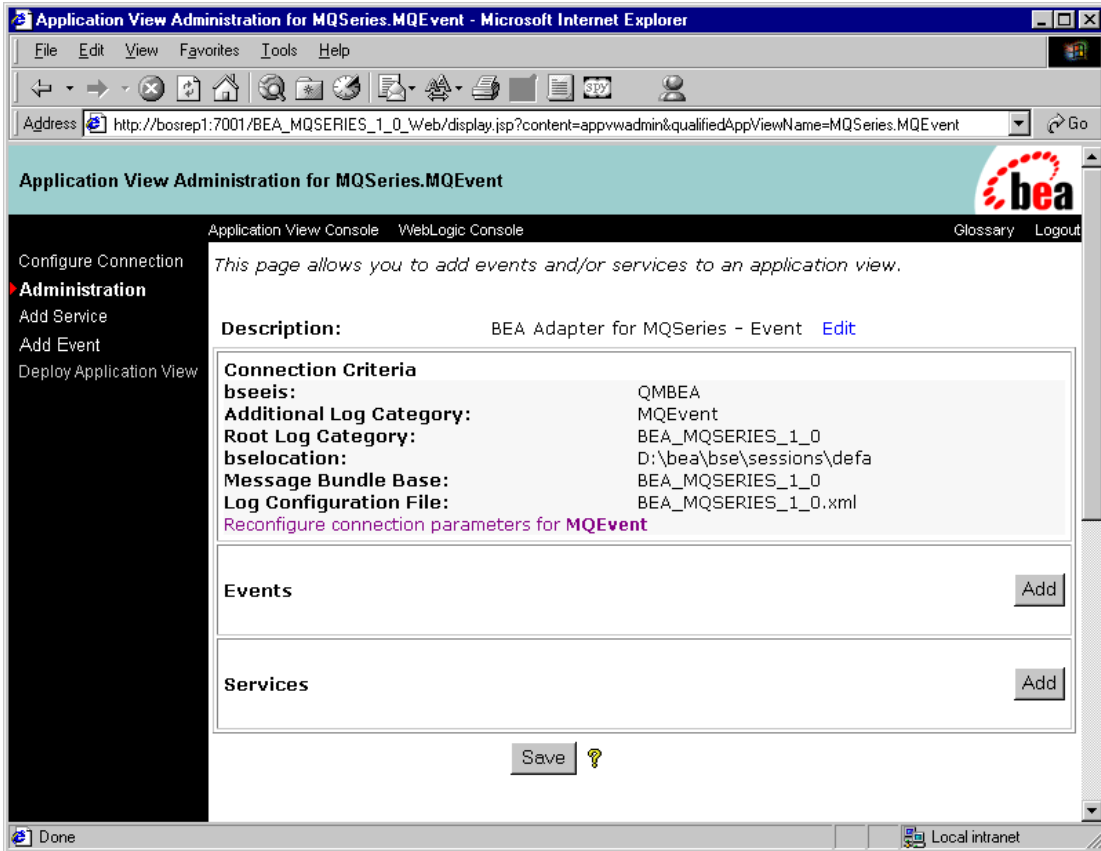
connection_name is the connection name you define (for example, MQConnect).

schema_files is where the schemas are stored. For more information about schema repositories, see [Chapter 2, “Metadata, Schemas, and Repositories.”](#)

10. Select the session name—also known as the connection name—from the Connection name drop-down list.
11. Click Connect to EIS. The Application View Administration window opens.

Note that you can access the Configure Connection Parameters window (displayed in the previous step) when the application view is not deployed, simply by clicking the Reconfigure connection parameters link. If the application view is deployed, you can access the window by first undeploying the application view.

Figure 3-6 Application View Administration Window



12. 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](#)” on page 3-10, before you can 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 schema should be added for each event to be used by the application view.

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

1. Log on to the WebLogic Application View Console as described in [“Creating an Application View Folder” on page 3-2](#).
2. Select the folder in which this application view resides, and then select the application view.

The Administration window opens.

3. Select Add Event in the left pane.

The Add Event window opens.

Figure 3-7 Add Event Window

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:* Order_excel

MQEvent

non-XML Preparse	excel(HAS_HEADERS)
XCH Transform	Order2OrderIn.xch
XSLT Transform	OrderIn2HTML.xml
Queue Manager*	QMBEA
Queue Name*	TEST.excelQ
Retry Duration	
Retry Interval	
Recycle Interval	
MQ Client Host	
MQ Client Port	
MQ Client Channel	
Execution Time Limit	
Polling Interval	

schema: OrderIn

Add

Done Local intranet

3 *Creating and Configuring an Event Adapter*

The properties in this window correspond to the MQSeries communication and transformation settings that the event adapter uses. The adapter uses these settings to communicate with MQSeries and to process messages (messages are also known as documents).

The schema drop-down list corresponds to the list of events in the schema repository.

Table 3-1 Event Properties

Property	Description	Type	Sample Value
non-XML Preparse	Name of the transform type selected for the non-XML to XML transformation phase. Supports entag, Excel, MFL, and .xch transform types. For more information, see Chapter 5, “Transforming Document Formats.”	string	excel
XCH Transform	Name of the .xch transform template file used by the XCH transform type in the XML to XML transformation phase. For more information, see Chapter 5, “Transforming Document Formats.”	string	
XSLT Transform	Name of the XSLT stylesheet used by the XSLT transform type in the XML to XML transformation phase. For more information, see Chapter 5, “Transforming Document Formats.”	string	
Queue Manager* (*Required)	Name of the MQSeries Queue Manager to be used.	string	OMBEA
Queue Name* (*Required)	Queue on which request documents are received.	string	TEST.iO
Retry Duration	Maximum time that a document can remain in the retry pending queue.	duration	
Retry Interval	Interval between retrying pending requests.	duration	
Recycle Interval	Interval between retrying successful requests.	duration	
MQ Client Host	For MQ Client only. Host on which MQ Server is located.	string	

Table 3-1 Event Properties

Property	Description	Type	Sample Value
MQ Client Port	For MQ Client only. Port number to connect to an MQ Server.	integer	
MQ Client Channel	For MQ Client only. Channel between an MQ Client and MQ Server.	string	
Execution Time Limit	The time a worker is allowed to execute before being terminated.	string	
Polling Interval	The maximum wait interval between checks for new documents. The higher this value, the longer the interval, and the fewer system resources that are used. The side effect of a high value is that the worker thread will not be able to respond to a stop command. If timeout is set to 0, the listener will run once and terminate. Default is 2 seconds.	string	Duration, in the format <i>nnH:nnM:nnS</i> For example, <i>1H:2M:3S</i> represents 1 hour 2 minutes and 3 seconds. (H, M, and S are not case sensitive.)

4. From the drop-down list, select the name of the schema that describes the event you are adding.
5. Click Add. The Application View Administration window opens.

3 Creating and Configuring an Event Adapter

6. Click Continue. The Deploy Application View window opens.

Figure 3-8 Deploy Application View Window

Deploy Application View MQSeries.MQEvent to Server - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://bosrep1:7001/BEA_MQSERIES_1_0_Web/display.jsp Go

Deploy Application View MQSeries.MQEvent 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 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 MQEvent using J2EE Security](#)

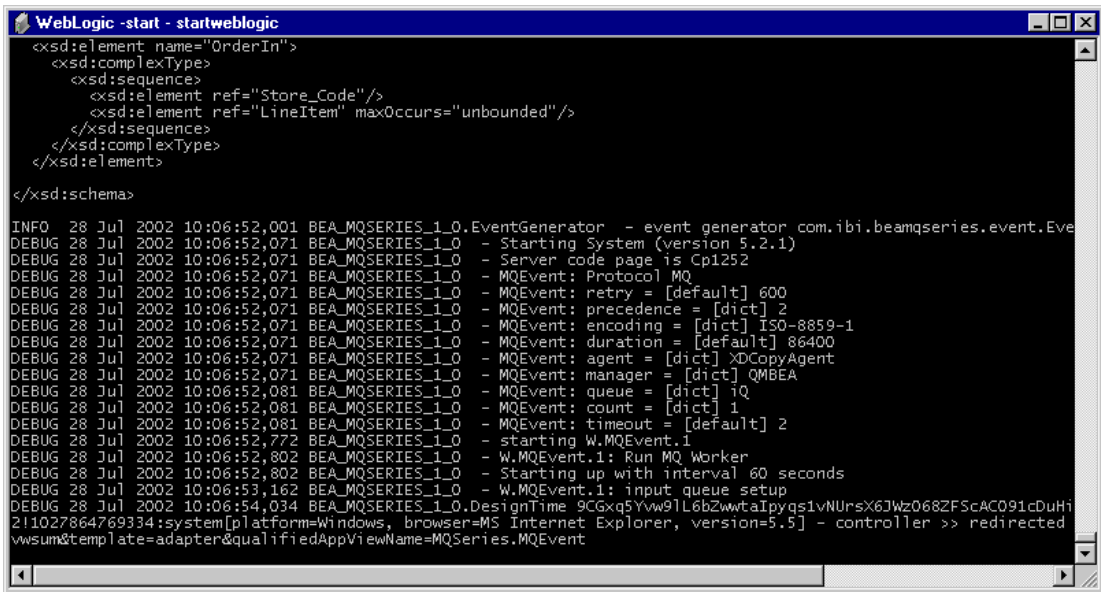
☒ Deploy persistently?

javascript:handleSecurity(); Local intranet

7. Update event parameters, connection pool parameters, log configuration, and security as necessary. For more information about these, see “Defining an Application View” in “Using Application Integration”:
 - For WebLogic Integration 7.0, see
<http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>
 - For WebLogic Integration 2.1, see
http://edocs.bea.com/wlintegration/v2_1sp/aiuser/2usrdef.htm
8. Click Deploy to save and deploy the event adapter.

In the WebLogic server log, you should see the following entries as the event adapter deploys.

Figure 3-9 WebLogic Server Log Window



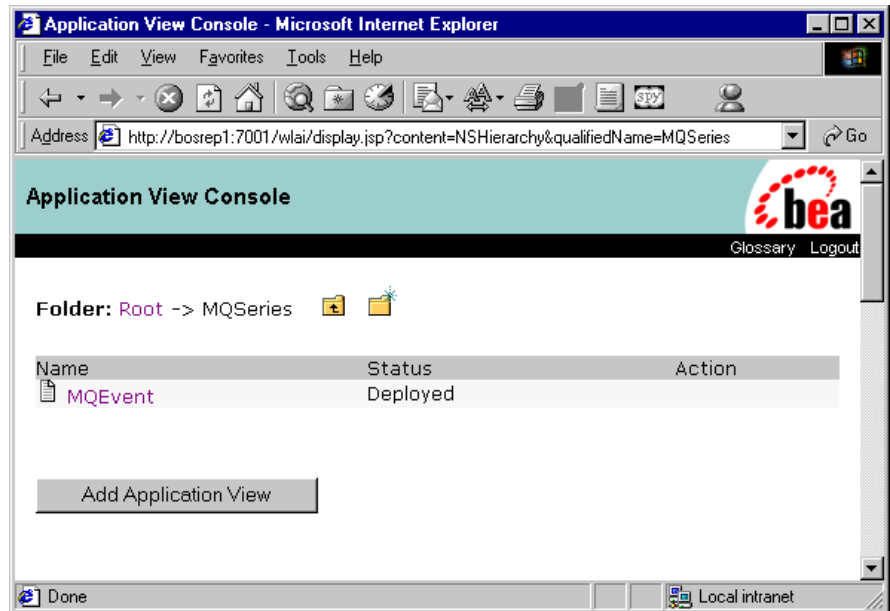
```
WebLogic -start -startweblogic
<xsd:element name="OrderIn">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Store_Code"/>
      <xsd:element ref="LineItem" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>

INFO 28 Jul 2002 10:06:52,001 BEA_MQSERIES_1_0.EventGenerator - event generator com.ibi.beamqseries.event.Eve
DEBUG 28 Jul 2002 10:06:52,071 BEA_MQSERIES_1_0 - Starting System (version 5.2.1)
DEBUG 28 Jul 2002 10:06:52,071 BEA_MQSERIES_1_0 - Server code page is Cp1252
DEBUG 28 Jul 2002 10:06:52,071 BEA_MQSERIES_1_0 - MQEvent: Protocol MQ
DEBUG 28 Jul 2002 10:06:52,071 BEA_MQSERIES_1_0 - MQEvent: retry = [default] 600
DEBUG 28 Jul 2002 10:06:52,071 BEA_MQSERIES_1_0 - MQEvent: precedence = [dict] 2
DEBUG 28 Jul 2002 10:06:52,071 BEA_MQSERIES_1_0 - MQEvent: encoding = [dict] ISO-8859-1
DEBUG 28 Jul 2002 10:06:52,071 BEA_MQSERIES_1_0 - MQEvent: duration = [default] 86400
DEBUG 28 Jul 2002 10:06:52,071 BEA_MQSERIES_1_0 - MQEvent: agent = [dict] XDCopyAgent
DEBUG 28 Jul 2002 10:06:52,071 BEA_MQSERIES_1_0 - MQEvent: manager = [dict] QMBEA
DEBUG 28 Jul 2002 10:06:52,071 BEA_MQSERIES_1_0 - MQEvent: queue = [dict] iQ
DEBUG 28 Jul 2002 10:06:52,081 BEA_MQSERIES_1_0 - MQEvent: count = [dict] 1
DEBUG 28 Jul 2002 10:06:52,081 BEA_MQSERIES_1_0 - MQEvent: timeout = [default] 2
DEBUG 28 Jul 2002 10:06:52,772 BEA_MQSERIES_1_0 - starting W.MQEvent.1
DEBUG 28 Jul 2002 10:06:52,802 BEA_MQSERIES_1_0 - W.MQEvent.1: Run MQ Worker
DEBUG 28 Jul 2002 10:06:52,802 BEA_MQSERIES_1_0 - Starting up with interval 60 seconds
DEBUG 28 Jul 2002 10:06:53,162 BEA_MQSERIES_1_0 - W.MQEvent.1: input queue setup
DEBUG 28 Jul 2002 10:06:54,034 BEA_MQSERIES_1_0.DesignTime 9CGxq5Yvw91L6bZwWtaIpyqs1vNursX6JWzO68ZFScAC091cDuHi
2!1027864769334:system[platform=Windows, browser=MS Internet Explorer, version=5.5] - controller >> redirected
vwsum&template=adapterqualifiedAppViewName=MQSeries.MQEvent
```

3 *Creating and Configuring an Event Adapter*

9. To validate that the application view was successfully deployed, go to the main Application View Console window, and select the folder in which you created the application view. You should see the name of the new application view.

Figure 3-10 Selecting MQEvent in the Main Window



You have finished configuring the event adapter application view.

You can confirm that you configured it correctly, and that it can successfully receive events, using the instructions in [“Testing Event Adapter Application Views Using Application View Console”](#) on page 3-17 and [“Testing Event Adapter Application Views Using WebLogic Integration Studio”](#) on page 3-24.

Testing Event Adapter Application Views Using Application View Console

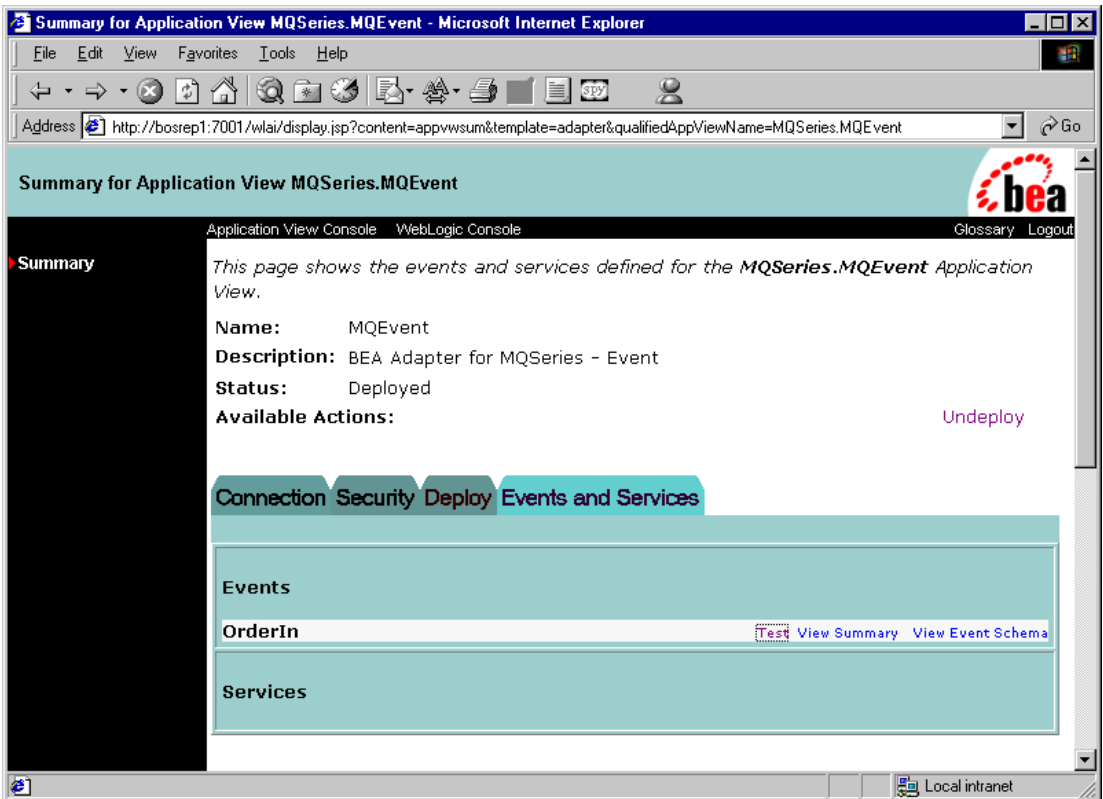
To confirm that a deployed event adapter application view is correctly configured and can receive events:

1. Log on to the WebLogic Application View Console as described in [“Creating an Application View Folder” on page 3-2](#).
2. Select the folder in which the application view resides, and then select the application view.

3 Creating and Configuring an Event Adapter

The Summary window opens.

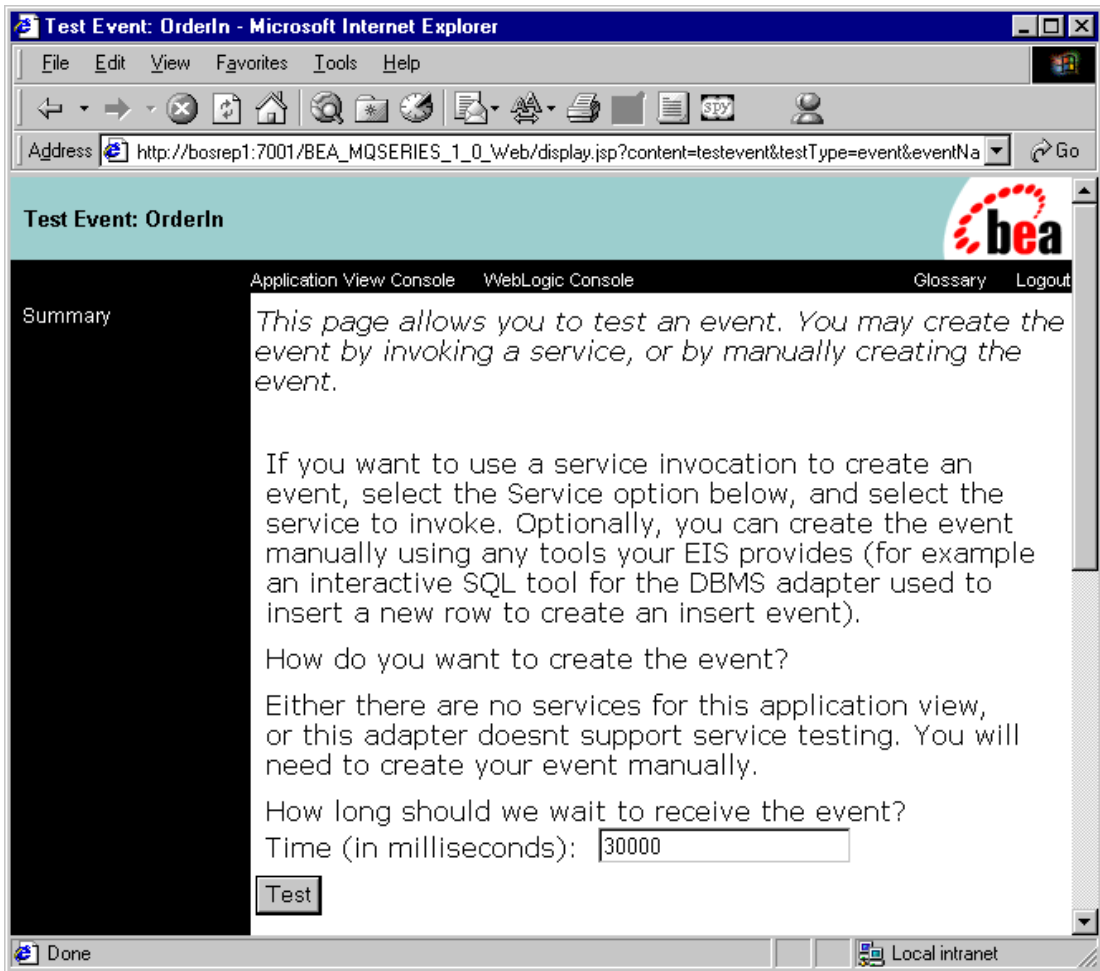
Figure 3-11 Summary Window



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

The Test Event window opens.

Figure 3-12 Test Event Window

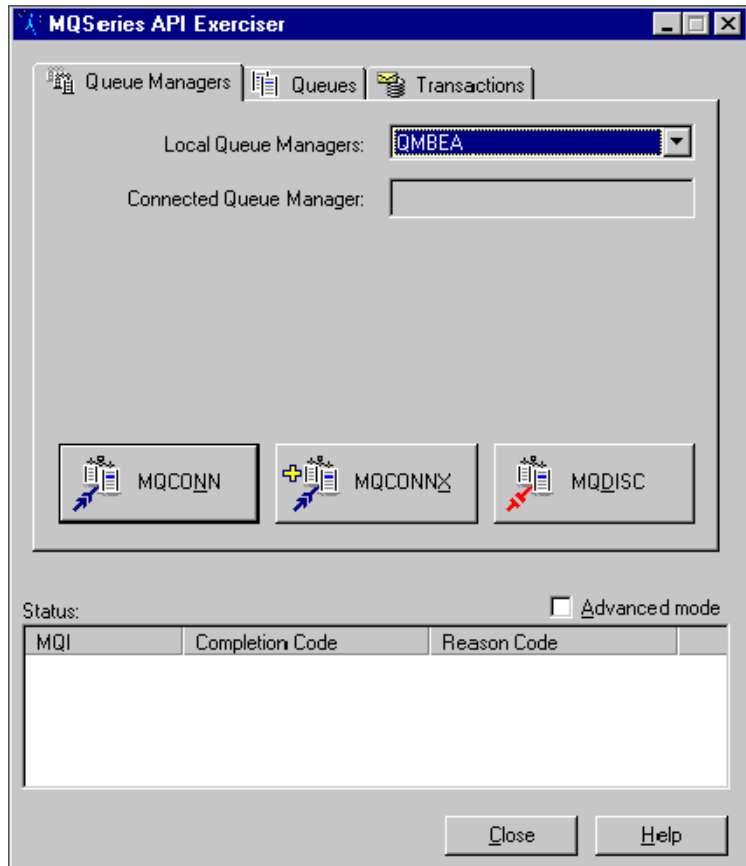


4. Enter 30000 (or a higher value) in the Time field. This provides a 30 second period during which, in the following step, you can access the IBM MQSeries API Exerciser (or your favorite utility) to manually invoke a request from MQSeries to your event adapter.

3 *Creating and Configuring an Event Adapter*

5. Open the MQSeries API Exerciser, and select the Queue Manager that you had assigned to the event adapter in the Application View Console's Add Event window.

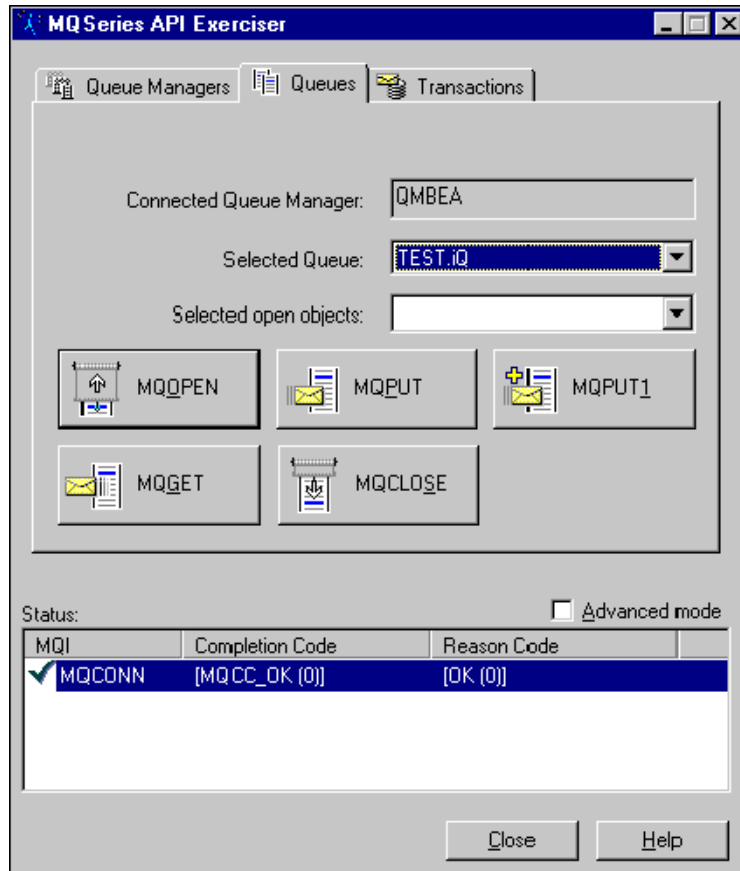
Figure 3-13 Queue Managers Tab in MQSeries API Exerciser Window



6. Click MQCONN.

- On the Queues tab, in the Selected Queue drop-down list, select the queue that you had assigned to the event adapter in the Application View Console's Add Event window.

Figure 3-14 MQSeries API Exerciser: Queues Tab

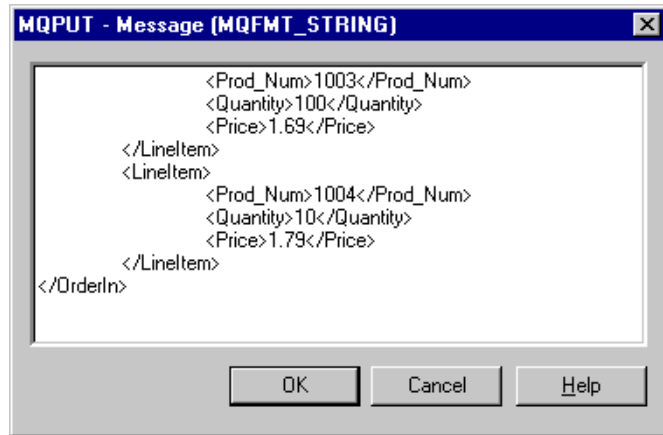


- Click MQOPEN.
- Click MQPUT. The MQPUT window opens.

3 *Creating and Configuring an Event Adapter*

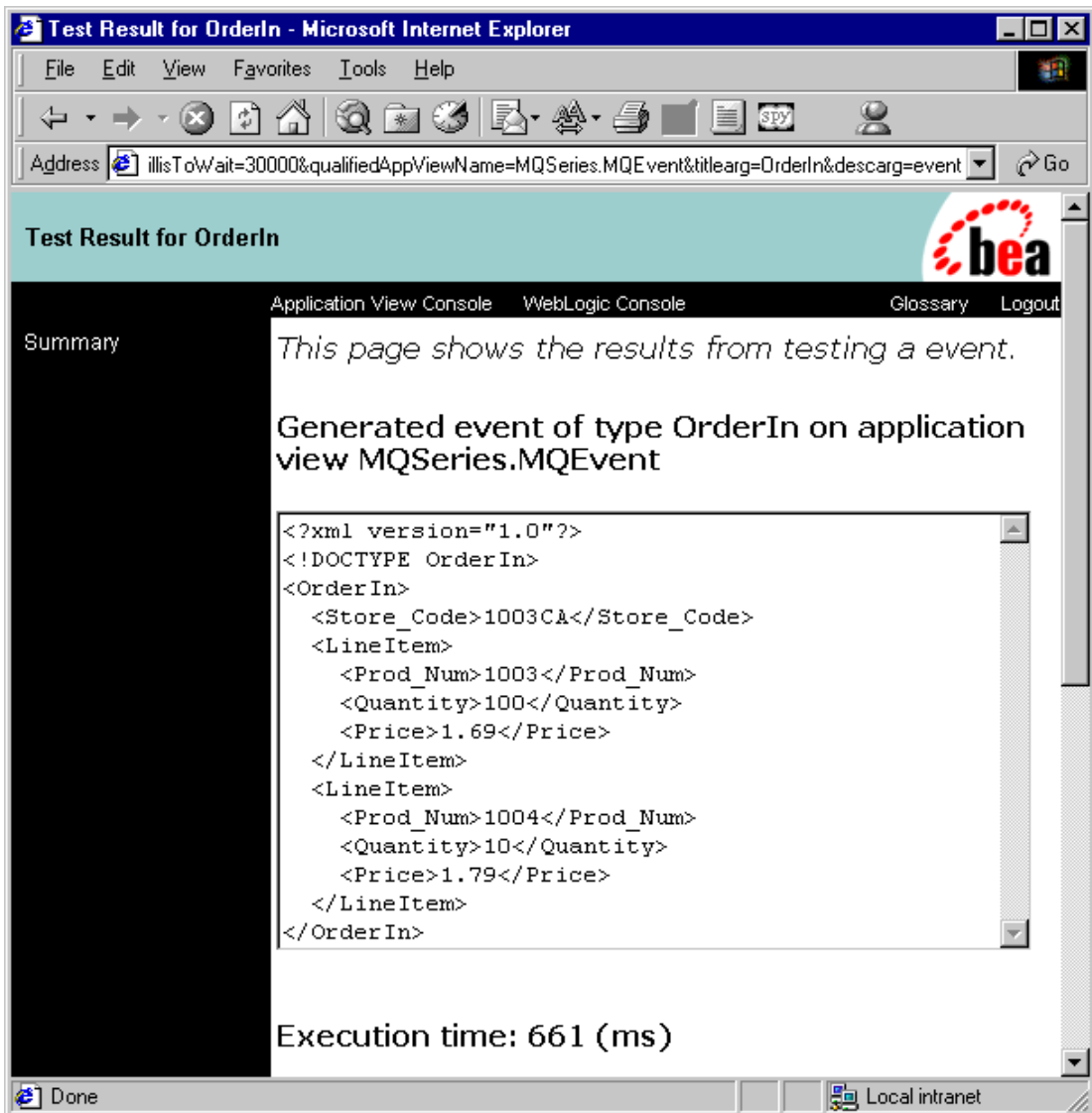
10. Copy and paste a sample instance document that matches the schema for the event that you are testing.

Figure 3-15 MQPUT Window



In the Application View Console, the Test Result window displays the event's result.

Figure 3-16 Test Result Window



If you wait longer than a minute and do not receive the event's result, you should assume that there is a problem with the event adapter application view. Examine the WebLogic Server log for information about the event's activity.

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

Testing Event Adapter Application Views Using WebLogic Integration Studio

To confirm that a deployed event adapter application view is correctly configured and can receive events:

1. Start the WebLogic Integration Studio:
 - On a Windows system, choose Start→Programs→BEA WebLogic Platform 7.0→WebLogic Integration 7.0→Studio.
 - On a UNIX system, go to the *WLI_HOME/bin* directory, and run the Studio start-up script by entering the following at the command prompt:

```
sh studio.sh
```

Here, *WLI_HOME* represents the root of your WebLogic Integration installation. For example:

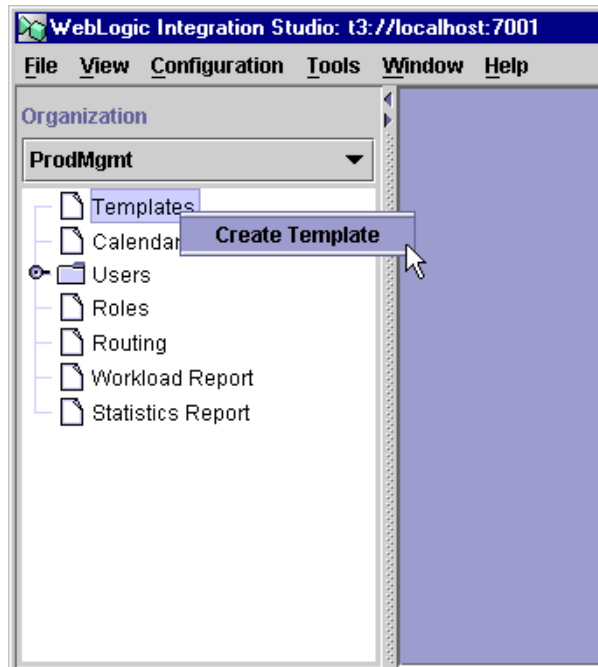
If you install WebLogic Integration 2.1 in the default location on a Windows system, *WLI_HOME* represents *c:\bea\wlintegration2.1*.

If you install WebLogic Integration 7.0 in the default location on a Windows system, *WLI_HOME* represents *c:\bea\weblogic700\integration*.

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

4. Right-click Templates and choose Create Template.

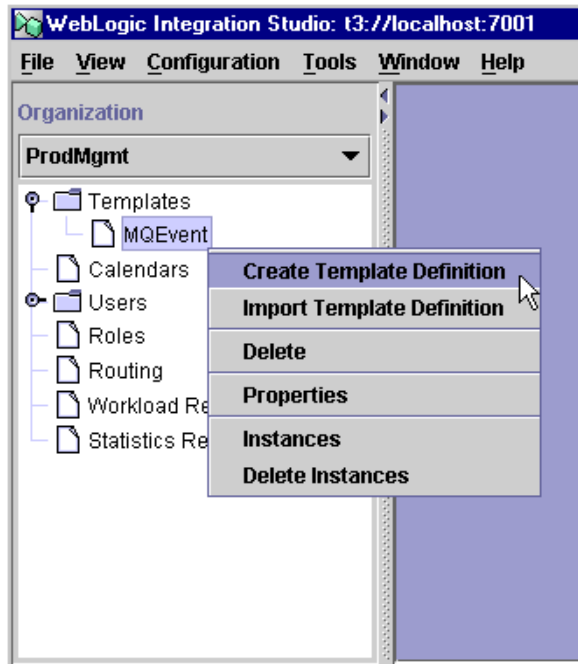
Figure 3-17 WebLogic Integration Studio



3 *Creating and Configuring an Event Adapter*

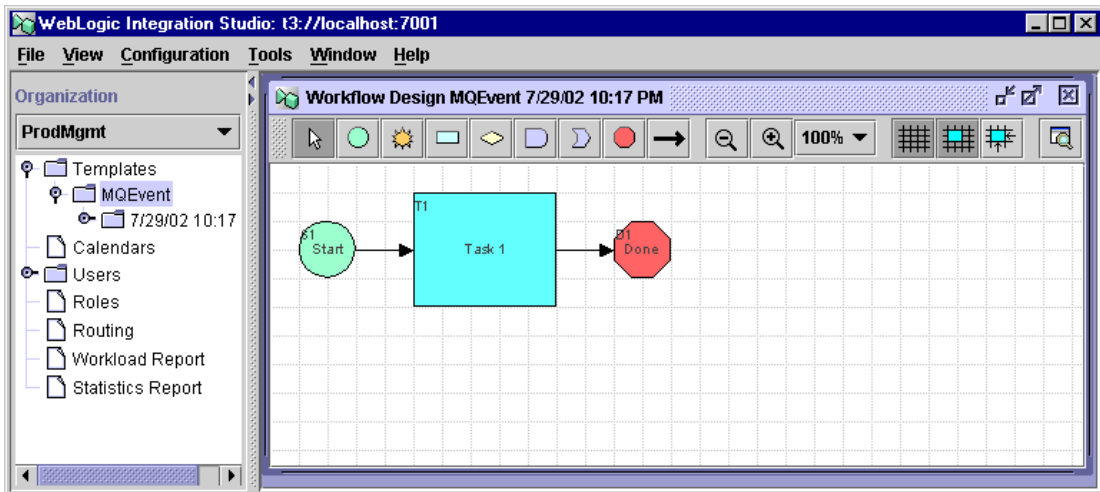
5. Right-click the new template and choose Create Template Definition.

Figure 3-18 Choosing Create Template Definition in WebLogic Integration Studio



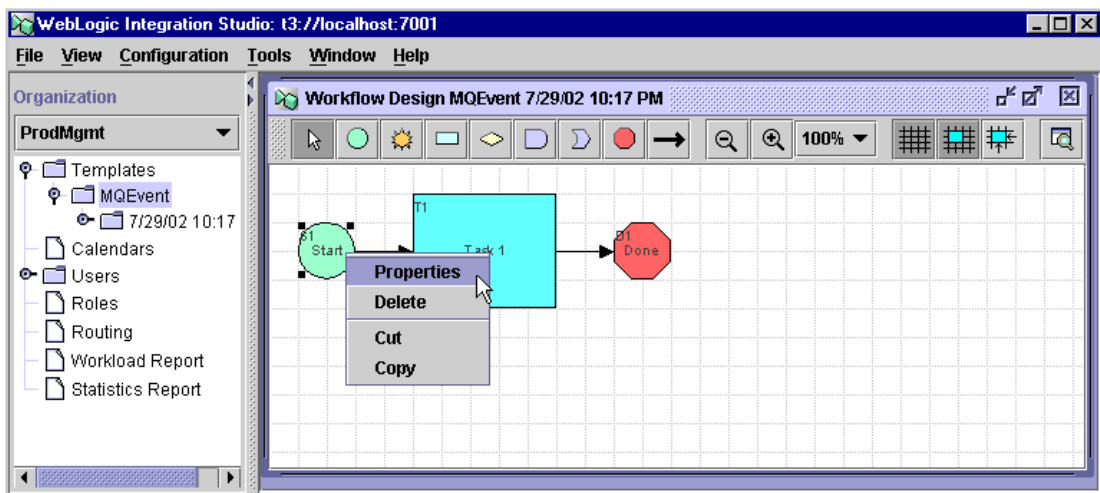
The template displays in WebLogic Integration Studio.

Figure 3-19 Displaying a New Template in WebLogic Integration Studio



6. Right-click the Start node and choose Properties.

Figure 3-20 Choosing Node Properties in WebLogic Integration Studio



3 *Creating and Configuring an Event Adapter*

The Start Properties window opens.

Figure 3-21 Start Properties Window

The screenshot shows the 'Start Properties' dialog box. It has a title bar with 'Start Properties' and a close button. The main area is divided into several sections:

- Description:** A text field containing 'Start'. Below it are radio buttons for 'Timed', 'Manual', 'Called', and 'Event' (which is selected). To the right is a dropdown menu showing 'AI Start'.
- Tree View:** A folder tree on the left showing a hierarchy: Root > MQSeries > MQTesting > MQEvent > OrderIn_csv (selected). Other folders include MQService and SAP.
- Properties:** On the right, there are fields for 'Name:' (containing 'OrderIn'), 'Description:', 'Condition:', and 'Event Document Variable:'. There is a search button 'A+BQ' next to the Condition field.
- Buttons:** 'Refresh Tree' and 'View Definition' buttons are located below the Event Document Variable field.
- Start Organization:** A dropdown menu and a checkbox labeled 'Use workflow expression'.
- Variables:** A tabbed interface with 'Variables', 'Actions', 'Next', and 'Notes' tabs. The 'Variables' tab is active, showing a table with columns 'Variable' and 'Expression'. To the right of the table are 'Add', 'Update', and 'Delete' buttons.
- Footer:** 'OK', 'Cancel', and 'Help' buttons.

7. Select Event→AI Start.
8. In the event explorer, browse the Application View folders and select the application view that corresponds to the event adapter.
9. Open the event adapter and select the desired event.

3 *Creating and Configuring an Event Adapter*

10. Select <new> from the Event Document Variable drop-down list.

Figure 3-22 New Document List in Start Properties Window

The screenshot shows the 'Start Properties' dialog box. The 'Description' section has a text field containing 'Start'. Below it are radio buttons for 'Timed', 'Manual', 'Called', and 'Event' (which is selected). To the right of the radio buttons is a dropdown menu showing 'All Start'. The main area is divided into two panes. The left pane shows a tree view with 'Root' at the top, followed by 'MQSeries', 'MQTesting', 'MQEvent', 'MQService', and 'SAP'. Under 'MQEvent', there is a file named 'OrderIn_csv'. The right pane contains fields for 'Name:' (containing 'OrderIn'), 'Description:', 'Condition:', and 'Event Document Variable:'. The 'Event Document Variable:' dropdown menu is open, showing a list with '<new>' selected. Below the main configuration area is a 'Start Organization' dropdown menu and a checkbox labeled 'Use workflow expression'. At the bottom, there are tabs for 'Variables', 'Actions', 'Next', and 'Notes'. The 'Variables' tab is active, showing a table with two columns: 'Variable' and 'Expression'. To the right of the table are buttons for 'Add', 'Update', and 'Delete'. At the very bottom of the dialog are 'OK', 'Cancel', and 'Help' buttons.

Start Properties

Description

Start

☐ Timed ☐ Manual ☐ Called ☒ Event **All Start**

Name: OrderIn

Description:

Condition:

Event Document Variable:

Start Organization

☐ Use workflow expression

Variables **Actions** **Next** **Notes**

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

Add

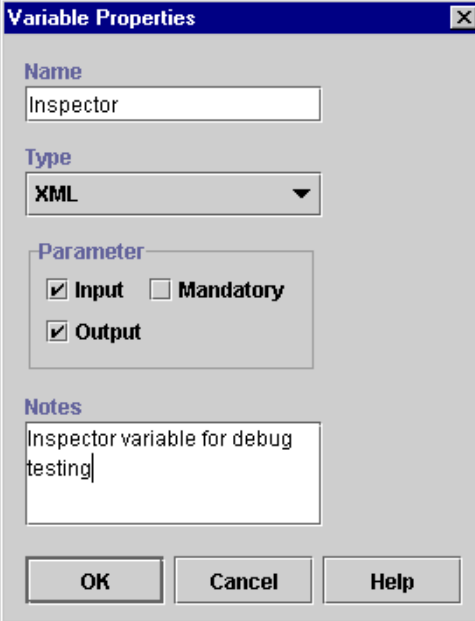
Update

Delete

OK **Cancel** **Help**

The Variable Properties window opens.

Figure 3-23 Variable Properties Window

The image shows a 'Variable Properties' dialog box with a blue title bar and a close button. It contains four sections: 'Name' with a text field containing 'Inspector'; 'Type' with a dropdown menu set to 'XML'; 'Parameter' with two checked checkboxes, 'Input' and 'Output', and an unchecked 'Mandatory' checkbox; and 'Notes' with a text area containing 'Inspector variable for debug testing'. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

Variable Properties

Name
Inspector

Type
XML

Parameter
☒ Input ☐ Mandatory
☒ Output

Notes
Inspector variable for debug testing

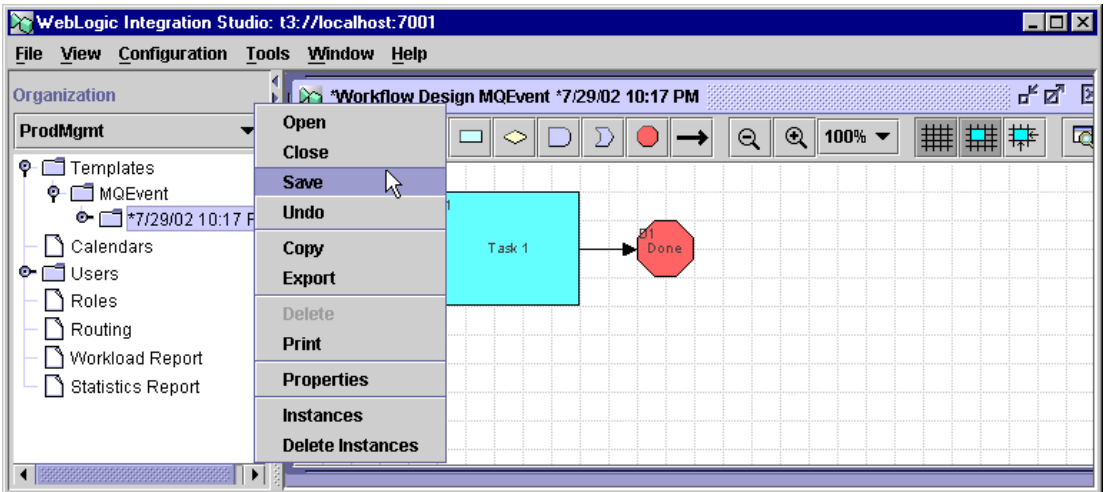
OK Cancel Help

11. Enter a name for the new variable, select the variable type XML, and select Input and Output in the Parameter group.
12. Click OK.

3 *Creating and Configuring an Event Adapter*

13. Right-click the template in WebLogic Integration Studio's Organization pane and choose Save.

Figure 3-24 Saving a Template in WebLogic Integration Studio



14. Right-click the event definition folder and choose Properties. The Template Definition window opens.

Figure 3-25 Template Definition Window

The screenshot shows a dialog box titled "Template Definition MQEvent" with a close button (X) in the top right corner. It has two tabs: "General" (selected) and "Exception Handlers". The "General" tab contains the following fields and controls:

- Workflow Label:** A text input field with a search icon (A+B) to its right.
- Active:** A checked checkbox.
- Effective:** A date dropdown menu showing "Jul 29, 2002".
- Expiry:** An unchecked checkbox next to a date dropdown menu showing "Dec 31, 2002".
- Enable auditing:** An unchecked checkbox.
- Notes:** A large text area for notes.

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

15. Ensure that Active is checked, and click OK.

You may now initiate events from your Enterprise Information System. For the BEA WebLogic Adapter for MQSeries, you can create events through a business application or through an MQSeries utility.

3 *Creating and Configuring an Event Adapter*

For example, you could use the PutMessage utility in MQSeries SupportPac MA0J to put messages into a queue to which the event adapter is configured.

Figure 3-26 MQSeries PutMessage Window

PutMessage 75 Successful

Queue Manager: QMBEA More

Queue Name: TEST.csv.iQ More

Hostname: More

Channel: More

Port: 1414 More

Disconnect Put Message in Queue View/Edit Header Options

☒ Get Message Body from File:

D:\My Documents\BEA\MQSeries\event_OrderIn.csv Browse

☐ Insert Message Text Manually:

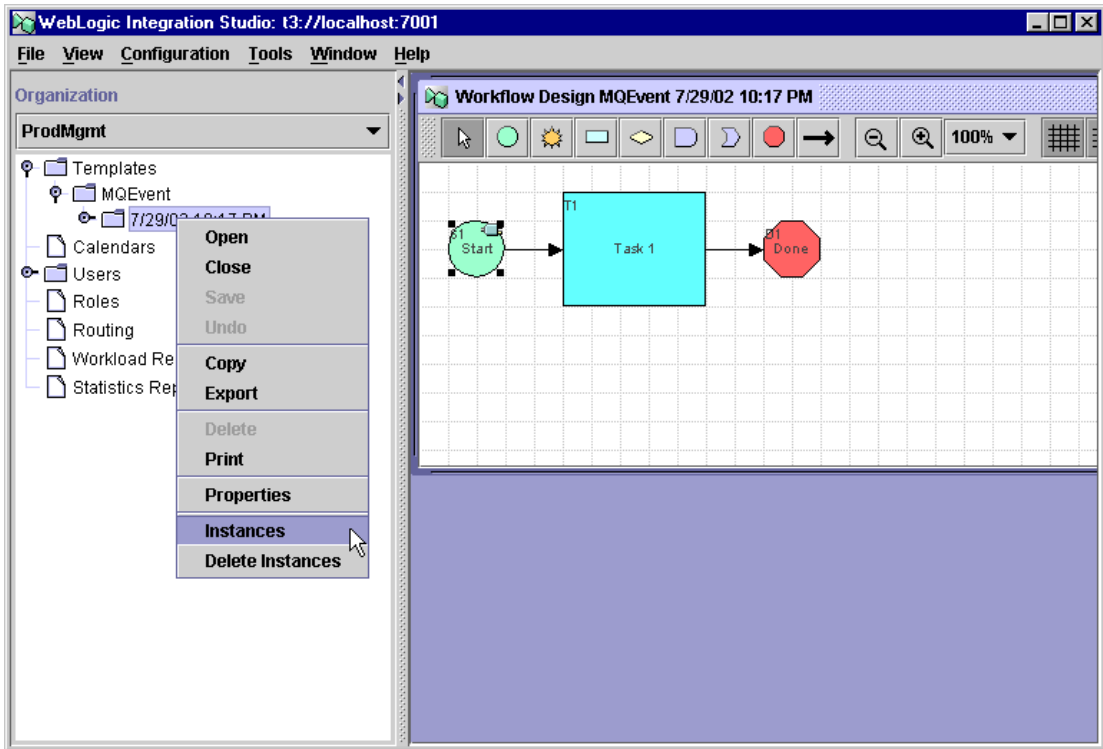
Close Help

This utility assists in writing messages to the queue. Either enter free-form text or source the data from a file.

Specify a Queue Manager, queue name, and port for your locally accessible MQ installation; select the message or message source file; and click Put Message in Queue.

16. Return to WebLogic Integration Studio.
17. Right-click the event definition folder and choose Instances.

Figure 3-27 Choosing Instances in WebLogic Integration Studio

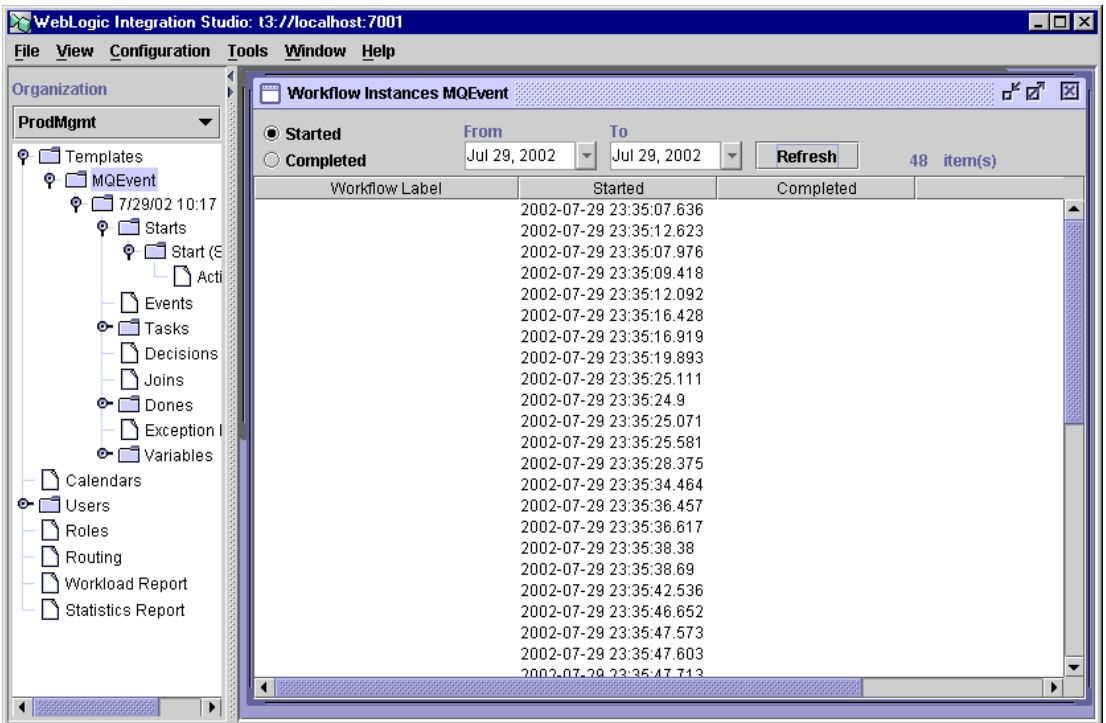


The Workflow Instances for your event definition appear. You can now track execution of your workflow.

3 *Creating and Configuring an Event Adapter*

18. Select Started and click Refresh. You will see a list of started workflows.

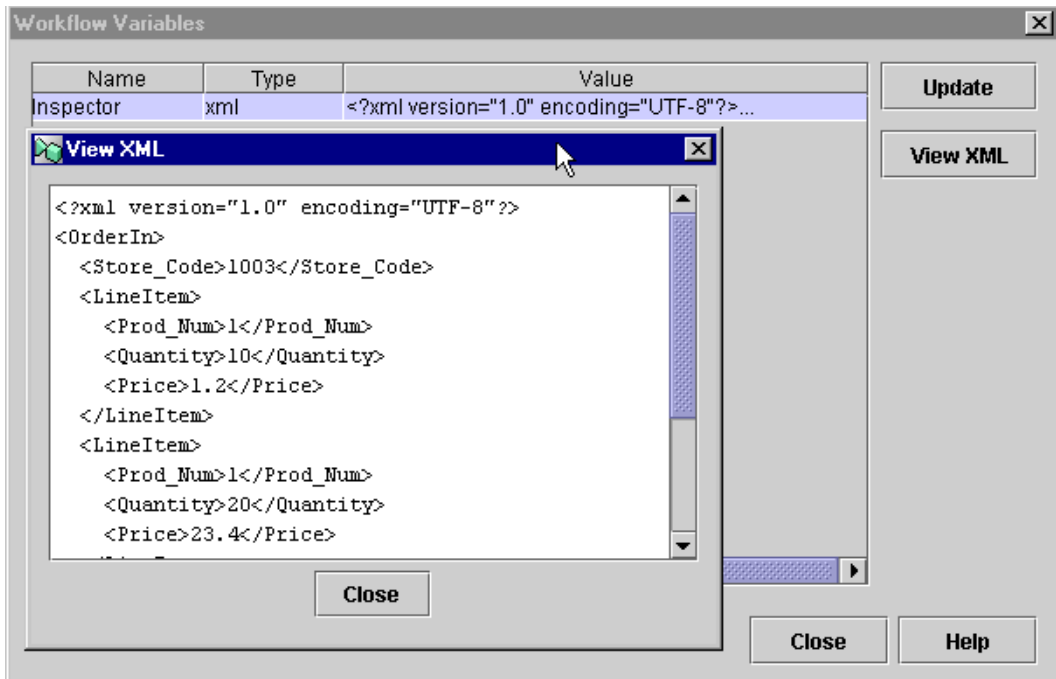
Figure 3-28 Displaying Workflow Instances in WebLogic Integration Studio



19. Right-click any instance of the workflow and choose Variables. The Workflow Variables window opens.

20. Click View XML to see the entire contents of the workflow message/document.

Figure 3-29 View XML Window



You have now confirmed that the event adapter application view is correctly configured and can receive events.

4 Creating and Configuring a Service Adapter

This section describes how to create, configure, and test a service adapter application view. The service adapter for MQSeries is WebLogic Integration's interface to IBM MQSeries, enabling your business processes to write to MQSeries queues. This section includes the following topics:

- [Creating a Service Adapter Application View](#)
- [Configuring a Service Adapter Application View](#)
- [Testing the BEA Service Adapter for MQSeries](#)

Creating a Service Adapter Application View

To create a service adapter application view:

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

Figure 4-1 Define New Application View Window

Application View Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address <http://bosrep1:7001/wlai/display.jsp?content=defappvw&namespa> Go

Define New Application View

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

This page allows you to define a new application view

Folder: **MQSeries**

Application View Name:*

Description:

Associated Adapter:

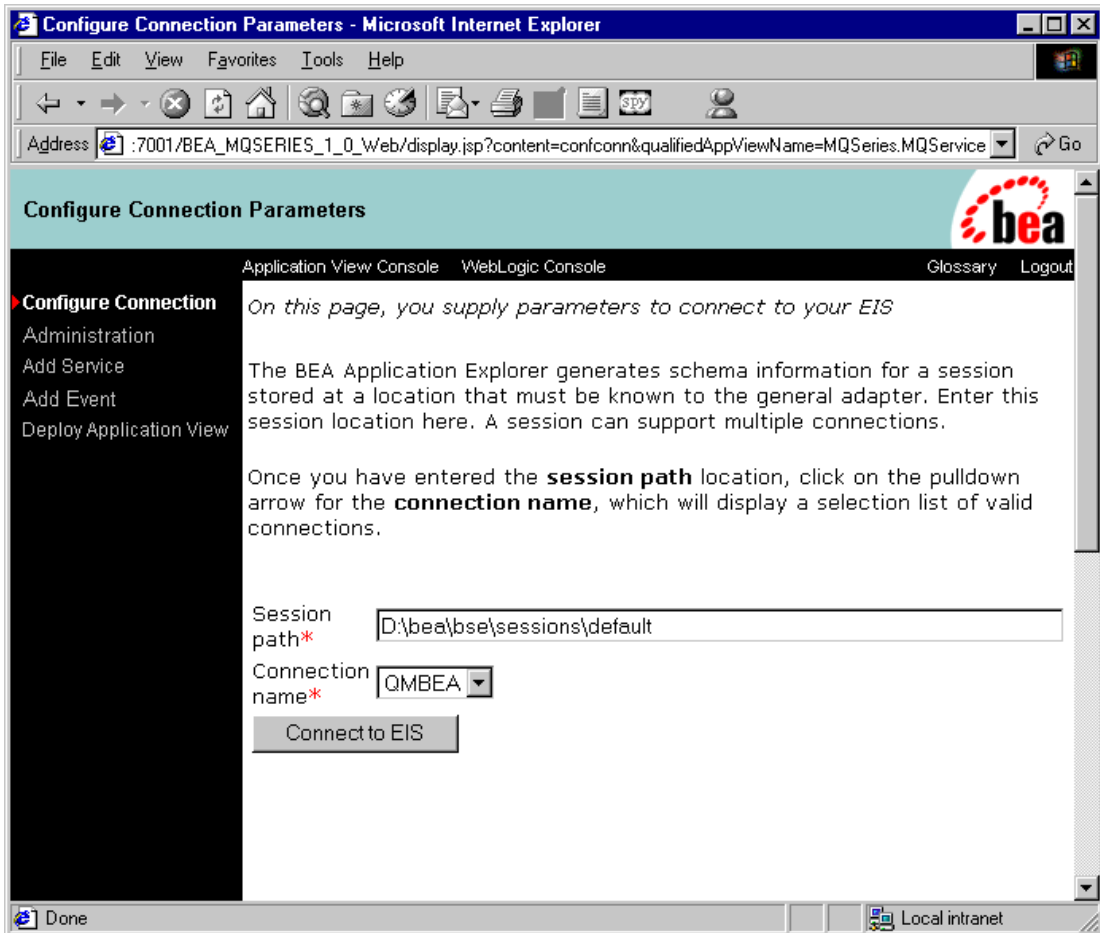
Done Local intranet

2. Enter a name and description for the application view, and select BEA_MQSERIES_1_0 from the Associated Adapter drop-down list.

3. Click OK.

The Configure Connection Parameters window opens.

Figure 4-2 Configure Connection Parameters Window



4. Enter the name of the BEA WebLogic Adapter for MQSeries session path (sometimes known as the session base directory).

This path holds your MQSeries schema and connection information:

drive:\session_path\mqseries\connection_name\schema_files

Here,

drive is the disk drive (for example, C).

session_path, which you are entering in this window, is the absolute path to the directory you wish to contain the directory structure for your XML schema files (for example, \BEA\Schemas).

connection_name is the connection name you define (for example, MQConnect).

schema_files is where the schemas are stored. For more information about schema repositories, see [Chapter 2, “Metadata, Schemas, and Repositories.”](#)

5. Select the session name—also known as the connection name—from the Connection name drop-down list.
6. Click Connect to EIS. The Application View Administration window opens.

Note that you can access the Configure Connection Parameters window (displayed in the previous step) when the application view is not deployed, simply by clicking the Reconfigure connection parameters link. If the application view is deployed, you can access the window by first undeploying the application view.

To configure a service adapter application view, see [“Configuring a Service Adapter Application View” on page 4-5](#)

Configuring a Service Adapter Application View

To configure a service adapter application view:

1. Log on to the WebLogic Application View Console at

`//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. If prompted, enter a user name and password.

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 MQSeries Installation and Configuration Guide*.

3. Click Login.

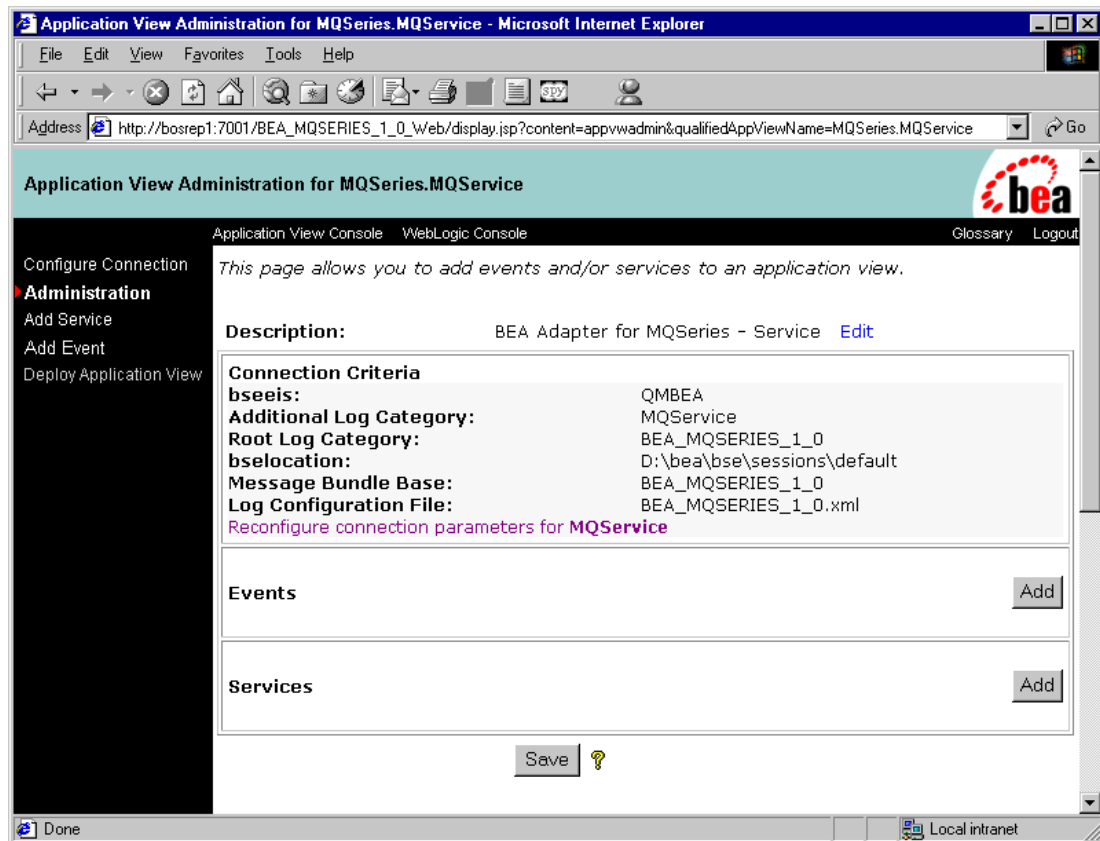
The WebLogic Integration Application View Console opens.

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

4 Creating and Configuring a Service Adapter

The Administration window opens.

Figure 4-3 Administration Window



5. Select Add Service in the left pane.

The Add Service window opens.

Figure 4-4 Add Service Window

Add Service - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://bosrep1:7001/BEA_MQSERIES_1_0_Web/display.jsp Go

Configure Connection
Administration
Add Service
Add Event
Deploy Application View

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

Unique Service Name*:

MQService

XCH_Transform	<input type="text"/>
XSLT_Transform	<input type="text"/>
non-XML_Pre-Emit	<input type="text"/>
Pre-Emit_Engine	<input type="text" value="mfl"/>
Queue_Manager*	<input type="text" value="QMBEA"/>
Queue_Name*	<input type="text" value="TEST.oQ"/>
Correlation_Id	<input type="text"/>
Retry_Duration	<input type="text"/>
Retry_Interval	<input type="text"/>
Recycle_Interval	<input type="text"/>
MQ_Client_Host	<input type="text"/>
MQ_Client_Port	<input type="text"/>
MQ_Client_Channel	<input type="text"/>
Execution_Time_Limit	<input type="text"/>
Polling_Interval	<input type="text"/>

schema:

settings

Trace_on/off ☐

Done Local intranet

6. Update the required properties of MQService as described in the following table.

These properties correspond to those that the service adapter uses to communicate with MQSeries to write messages to the queues and with the transform options described in [Chapter 5, “Transforming Document Formats.”](#)

Table 4-1 Service Properties

Property	Description	Type	Sample Value
XCH_Transform	Name of the .xch transform template file used by the XCH transform type in the XML to XML transformation phase. For more information, see Chapter 5, “Transforming Document Formats.”	string	
XSLT_Transform	Name of the XSLT stylesheet used by the XSLT transform type in the XML to XML transformation phase. For more information, see Chapter 5, “Transforming Document Formats.”	string	
non-XML_Pre-Emit	Name of the transform (pre-emit) parameter: either an MFL routine or an .xch transform file. Do not enter an .mfl extension for an MFL routine; these files are not stored with an extension. For more information, see Chapter 5, “Transforming Document Formats.”	string	XMLtoCSV.xch
Pre-Emit_Engine	Name of the transform type selected for the non-XML to XML transformation phase. Supports MFL and .xch transform types. For more information, see Chapter 5, “Transforming Document Formats.”	string	mfl

Table 4-1 Service Properties

Property	Description	Type	Sample Value
Queue_Manager* (Required)	Name of the MQSeries Queue Manager to be used.	string	OMBEA
Queue Name* (*Required)	Queue on which request documents are received.	string	TEST.io
Correlation_Id	The correlation ID to set in the MQSeries message header.	duration	
Retry_Duration	Maximum time that a document can remain in the retry pending queue.	duration	
Retry_Interval	Interval between retrying pending requests.	duration	
Recycle_Interval	Interval between retrying successful requests.	duration	
MQ_Client_Host	For MQ Client only. Host on which MQ Server is located.	string	
MQ_Client_Port	For MQ Client only. Port number to connect to an MQ Server.	integer	
MQ_Client_Channel	For MQ Client only. Channel between an MQ Client and MQ Server.	string	
Execution_Time_Limit	The time a worker is allowed to execute before being terminated.	string	

Table 4-1 Service Properties

Property	Description	Type	Sample Value
Polling_Interval	The maximum wait interval between checks for new documents. The higher this value, the longer the interval, and the fewer system resources that are used. The side effect of a high value is that the worker thread will not be able to respond to a stop command. If timeout is set to 0, the listener will run once and terminate. Default is 2 seconds.	string	Duration, in the format <i>nnH:nnM:nnS</i> . For example, 1H:2M:3S represents 1 hour 2 minutes and 3 seconds. (H, M, and S are not case sensitive.)

The schema drop-down list corresponds to the manifest that describes all event schemas.

7. Click Add.

The Administration window opens.

8. Click Continue.

The Deploy Application View window opens.

Figure 4-5 Deploy Application View Window

Deploy Application View MQSeries.MQService to Server

File Edit View Favorites Tools Help

Address http://bosrep1:7001/BEA_MQSERIES_1_0_Web/display.jsp Go

Deploy Application View MQSeries.MQService 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? ☒

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 MQService using J2EE Security](#)

☒ Deploy persistently?

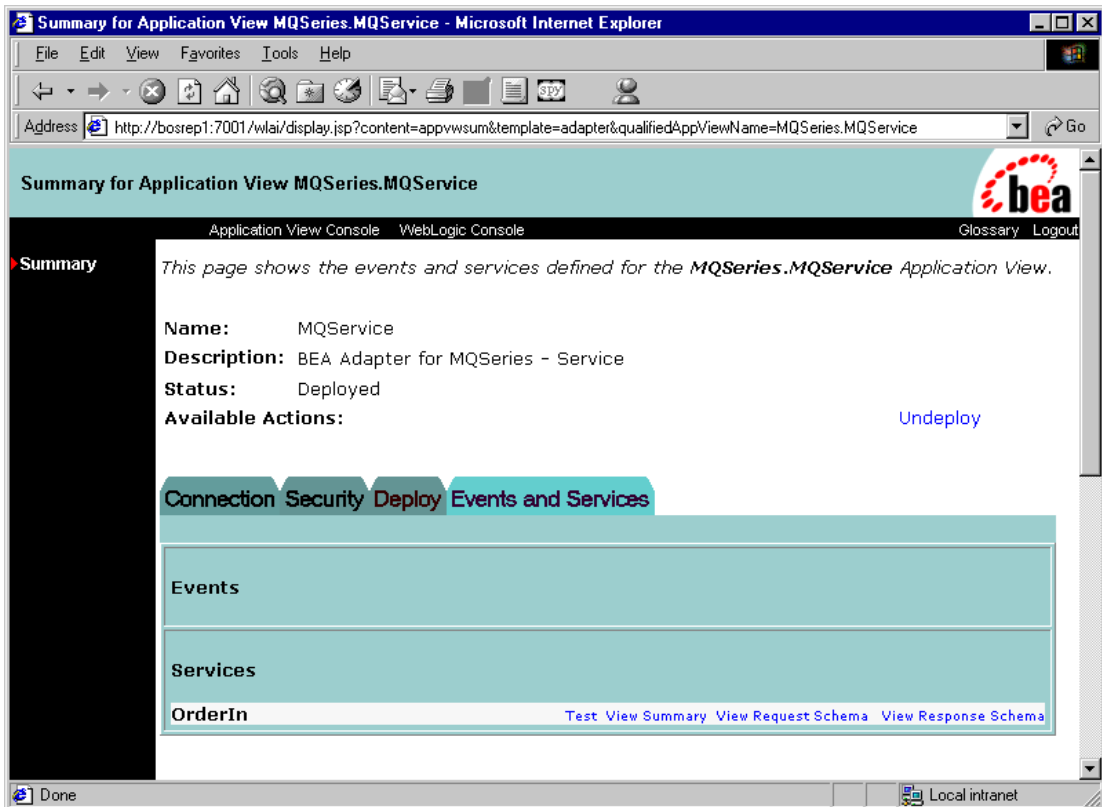
Done Local intranet

9. Update service parameters, connection pool parameters, log configuration, and security as necessary. For more information about these, see “Defining an Application View” in “Using Application Integration”:
 - For WebLogic Integration 7.0, see
<http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>
 - For WebLogic Integration 2.1, see
http://edocs.bea.com/wlintegration/v2_1sp/aiuser/2usrdef.htm

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

The Summary for Application View MQSeries.MQService window opens upon successful deployment.

Figure 4-6 Summary for Application View Window



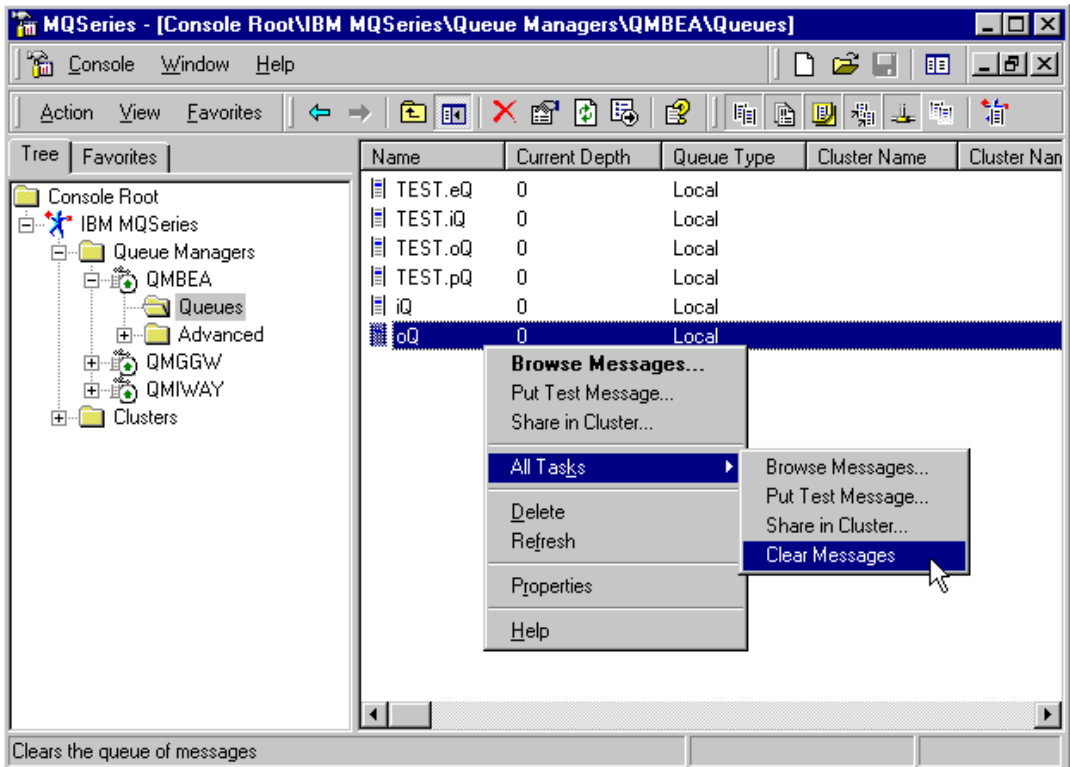
You have finished configuring the service adapter application view. You can confirm that you configured it correctly by following the instructions in [“Testing the BEA Service Adapter for MQSeries”](#) on page 4-14.

Testing the BEA Service Adapter for MQSeries

The service adapter emits a document to MQSeries and returns an emit status. You can validate the document's arrival on the queue by browsing the MQSeries resources through IBM-supplied or custom tools. For example, on a Windows platform, you can use MQSeries Explorer, a Microsoft MMC plug-in utility. Using this tool to connect to and explore the queues on the Queue Manager, you can view the state of the queue and browse the messages in it.

1. Confirm that the queue to which the service adapter sends documents is empty. For example, using MQSeries Explorer, browse the applicable queue and check the current queue depth, ensuring that it is zero (you can delete the existing queue messages by right-clicking and selecting All Tasks and then Clear Messages).

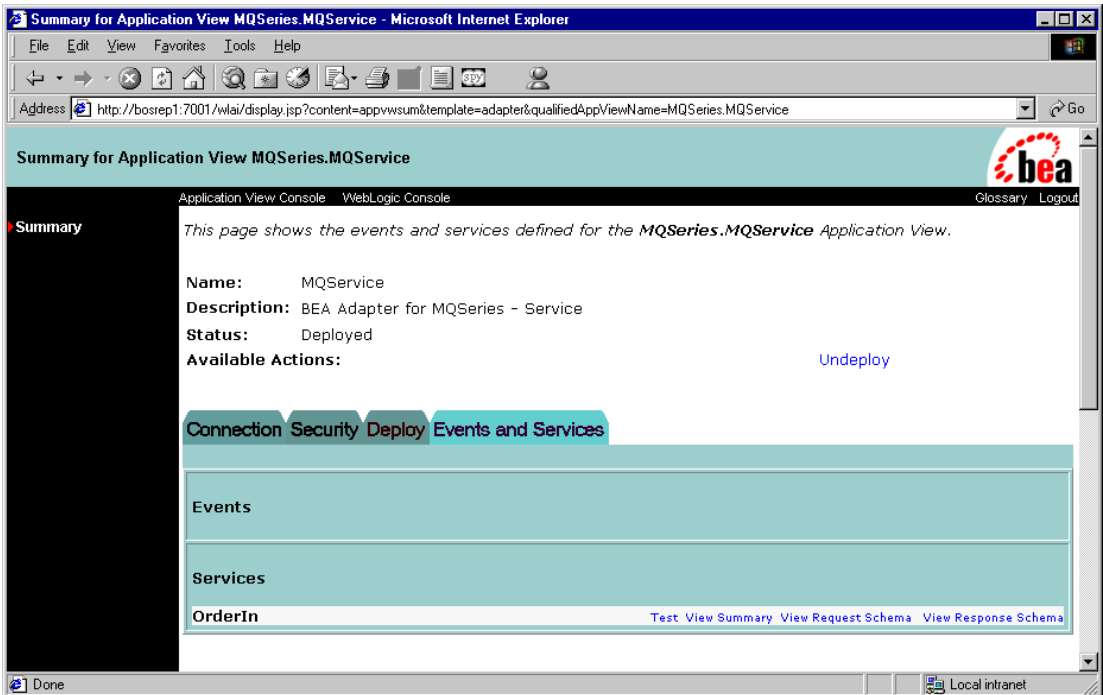
Figure 4-7 MQSeries Explorer



4 Creating and Configuring a Service Adapter

2. In the Application View Console's Summary for Application View window, click Test.

Figure 4-8 Testing the Service Using the Summary for Application View Window



3. Enter a sample document that matches the request schema for the configured service. For example, the `OrderIn` request schema has an instance document like the following code listing.

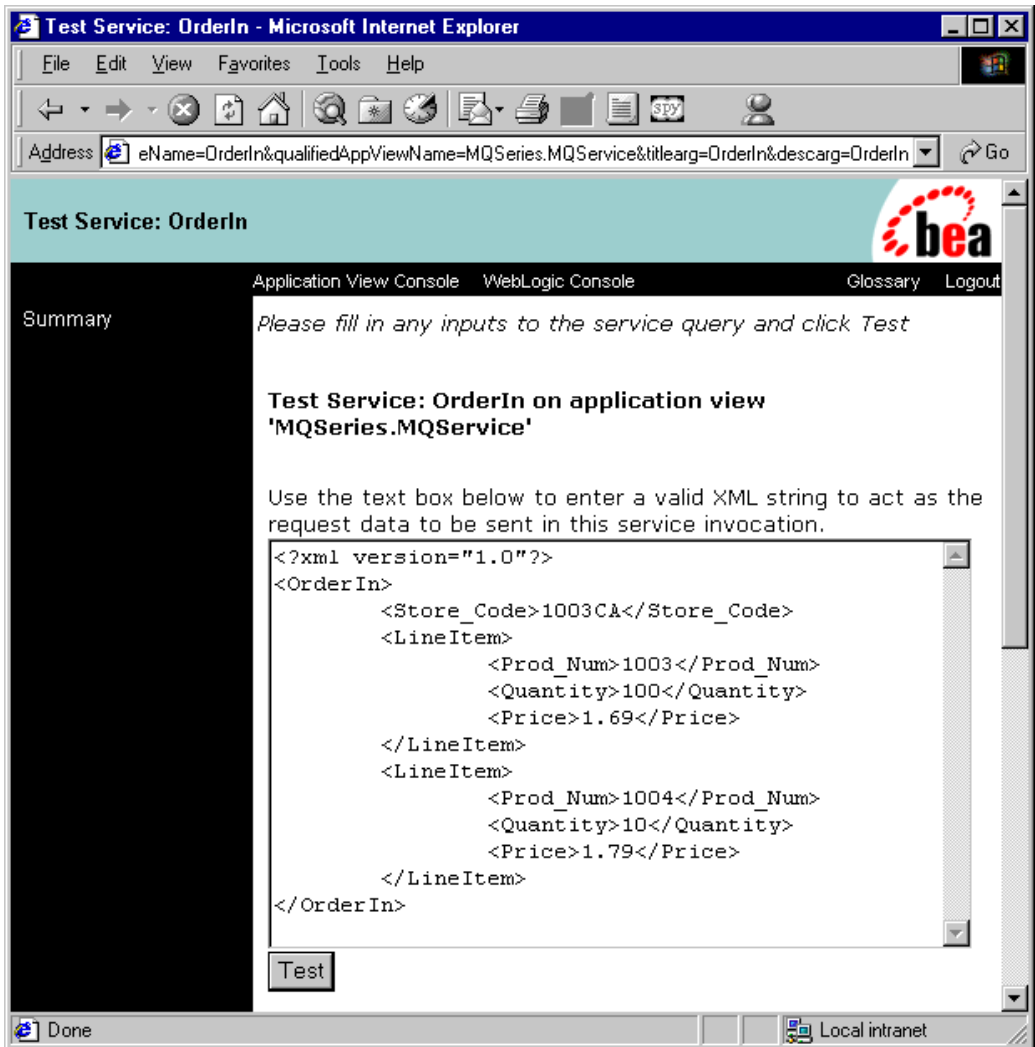
Listing 4-1 Instance Document for OrderIn Request Schema

```
<?xml version="1.0"?>

<OrderIn>
  <Store_Code>1003CA</Store_Code>
  <LineItem>
    <Prod_Num>1003</Prod_Num>
    <Quantity>100</Quantity>
    <Price>1.69</Price>
  </LineItem>
  <LineItem>
    <Prod_Num>1004</Prod_Num>
    <Quantity>10</Quantity>
    <Price>1.79</Price>
  </LineItem>
</OrderIn>
```

4. Enter this document into the Test Service window, by either typing it or copying and pasting it into the window.

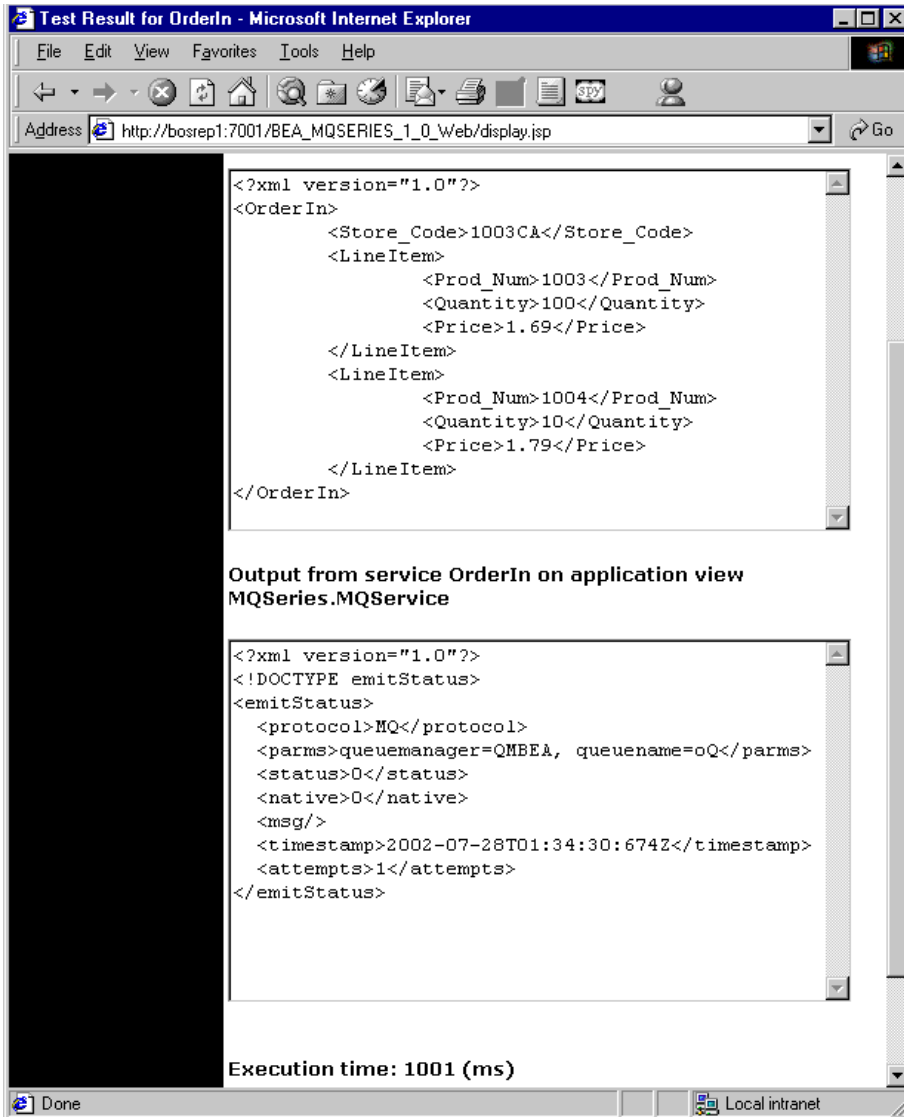
Figure 4-9 Entering OrderIn into the Test Service Window



5. Click Test to send the request through the service adapter to the IBM MQSeries queue.

The response document should indicate that the request was successfully emitted to MQSeries. See the sample response in the following figure.

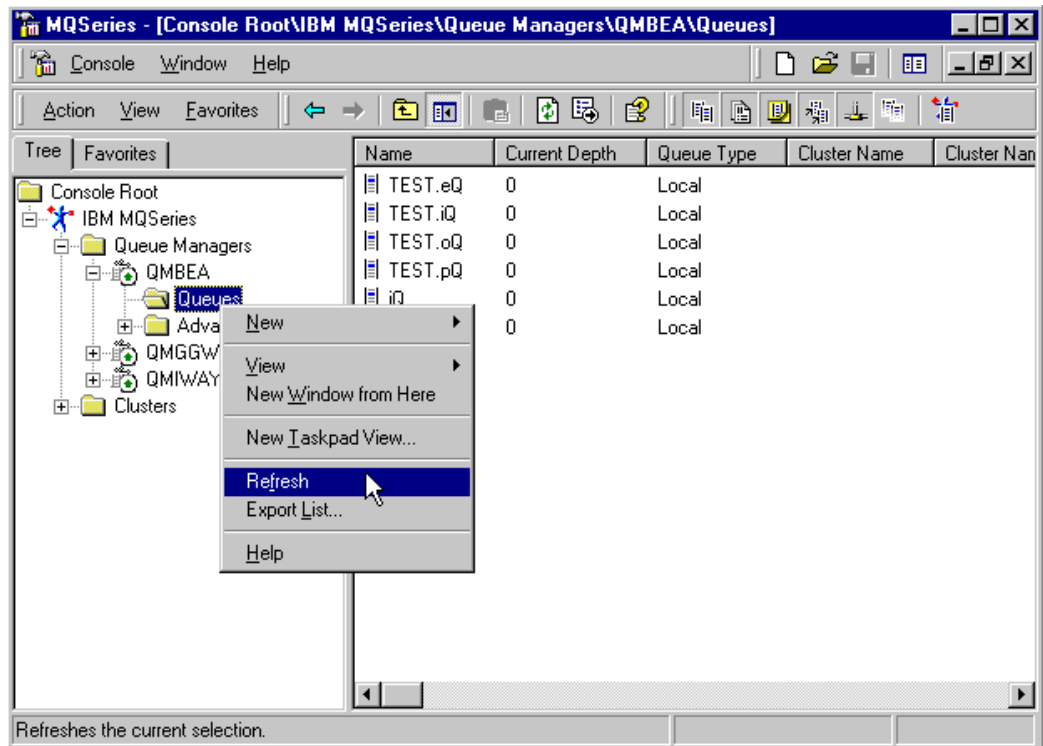
Figure 4-10 Test Result Window



4 Creating and Configuring a Service Adapter

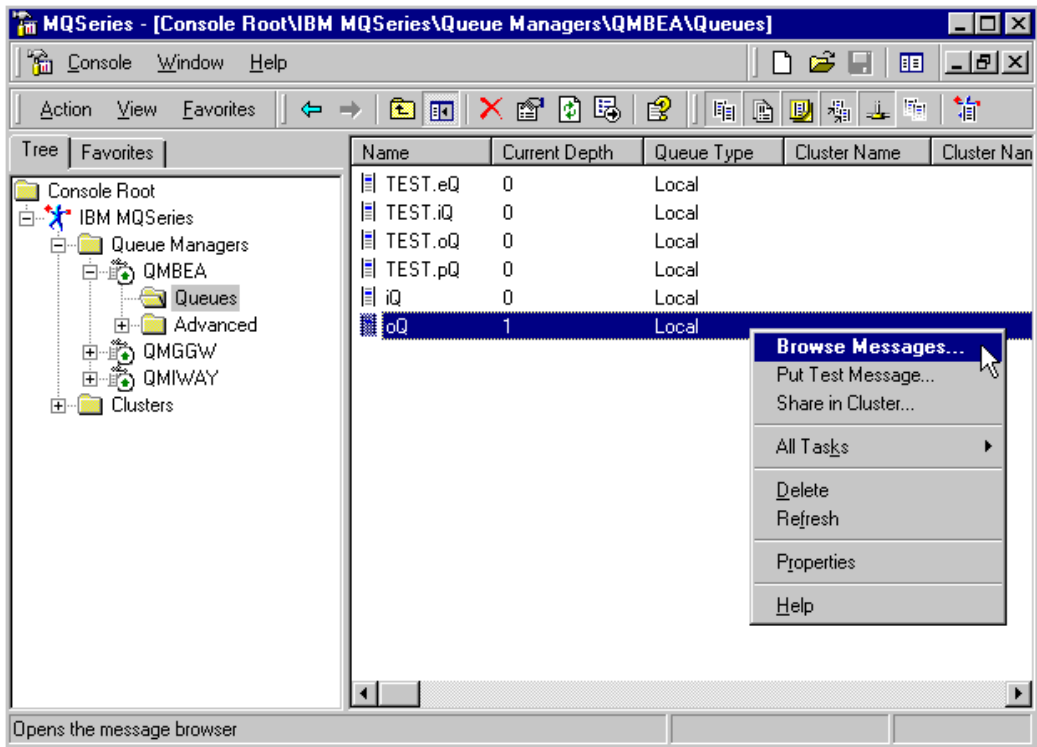
6. After any transformation and formatting, the document should have arrived on the MQSeries queue. Check for the output document on the queue designated in configuring the service. Using MQSeries Explorer, browse to the appropriate queue, and choose Refresh.

Figure 4-11 Refreshing the Queue in MQSeries Explorer



7. The current depth should be increased by one (there should be one item on the queue if you cleared all messages first). Right-click the appropriate queue and choose Browse Messages.

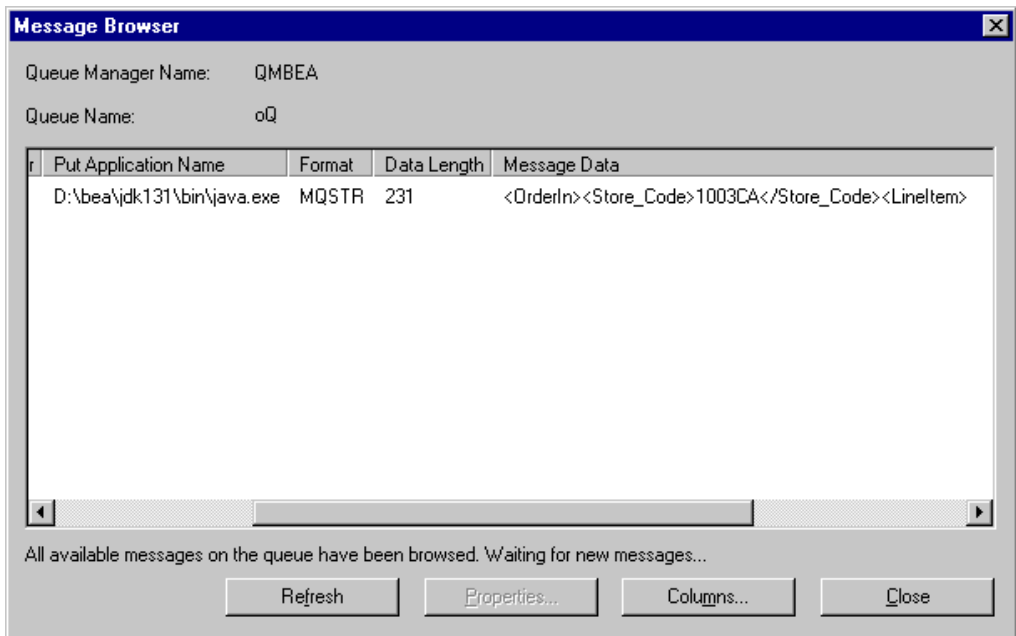
Figure 4-12 Browsing Messages in MQSeries Explorer



4 *Creating and Configuring a Service Adapter*

The Message Browser window, with a column containing the data sent by the service adapter to MQSeries, is displayed.

Figure 4-13 Message Browser Window



5 Transforming Document Formats

Documents within WebLogic Integration are encoded in XML. However, you may also need to receive and generate non-XML data. The BEA WebLogic Adapter for MQSeries supports inbound transformations for the event adapter and outbound transformations for the service adapter. This section describes the transformation options available to you. It includes the following topics:

- [Kinds of Transformations](#)
- [Creating MFL Transformations](#)
- [Testing MFL Transformations](#)

Kinds of Transformations

WebLogic Integration supports several transformation phases for converting data from one format to another. Each phase offers several methods, or transforms, for accomplishing the conversion.

There are two transformation phases available to an event adapter sending a message from MQSeries to a workflow. You can invoke either phase or both phases in sequence:

1. Non-XML formats to XML.
2. XML to XML.

There are two transformation phases available to a service adapter sending a message from a workflow to MQSeries. You can invoke either phase, or both phases in sequence:

1. XML to XML.
2. XML to non-XML formats.

Specify the type of transformation when adding an event or service to an application view.

Non-XML to XML Transformation

An event adapter that deals with non-XML data sources must be able to convert that data to XML for processing by a workflow. This conversion entails “pre-parsing” the data into XML, which is then parsed itself for processing by the workflow.

You can choose between three types of transforms—also known as pre-parsers—to accomplish this conversion:

- *entag*, which brackets a document with any XML element you specify, enabling you to import a non-XML document into an XML system.
- *Excel*, which converts documents in Excel format to XML.
- *MFL (Message Format Language)*, which uses BEA routines.

For more information about MFL based transformations, see “Building Format Definitions” in *Translating Data*:

- For WebLogic Integration 7.0, see
<http://edocs.bea.com/wli/docs70/diuser/fmtdef.htm>
- For WebLogic Integration 2.1, see
http://edocs.bea.com/wlintegration/v2_1sp/diuser/fmtdef.htm
- *XCH*, a general conversion system which uses dictionaries (implemented as `.dic` files) and transformation templates (implemented as `.xch` files).

You can invoke only one transform during this phase—that is, an event adapter can invoke only one kind of non-XML to XML conversion.

For example, in the following figure, the `Order_excel` event specifies the Excel transform for the non-XML to XML phase. Note that it also specifies two transforms—XCH and XSLT—for the XML to XML phase, which is described in [“XML to XML Transformations” on page 5-6](#).

Figure 5-1 Specifying Non-XML to XML Transform in Add Event Window

Add Event - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://bosrep1:7001/BEA_MQSERIES_1_0_Web/display.jsp Go

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*:

MQEvent

non-XML Preparse	<input type="text" value="excel(HAS_HEADERS)"/>
XCH Transform	<input type="text" value="Order2OrderIn.xch"/>
XSLT Transform	<input type="text" value="OrderIn2HTML.xsl"/>
Queue Manager*	<input type="text" value="QMBEA"/>
Queue Name*	<input type="text" value="TEST.excelQ"/>
Retry Duration	<input type="text"/>
Retry Interval	<input type="text"/>
Recycle Interval	<input type="text"/>
MQ Client Host	<input type="text"/>
MQ Client Port	<input type="text"/>
MQ Client Channel	<input type="text"/>
Execution Time Limit	<input type="text"/>
Polling Interval	<input type="text"/>

schema:

Done Local intranet

When you specify non-XML to XML transformation in the WebLogic Application View Console's Add Event window, use this syntax:

```
transformType(parameter_1, ..., parameter_n)
```

The parameters for each type of non-XML to XML transform (that is, for each pre-parser) are shown in the following table.

Table 5-1 Non-XML Pre-Parse Transform Parameters

Transform Type	Parameter	Value/Description/Example
entag	tag name	<p>Type/Value: string literal</p> <p>Description: The name of the XML element whose tags bracket a non-XML document, enabling you to import the document into an XML system.</p> <p>Example: entag(invoice) This produces an XML document of the form:</p> <pre><invoice> John Smith, 123 Main Street, NY, NY, 10121, \$100.00 </invoice></pre>
excel	HAS_HEADERS	<p>Type/Value: string literal</p> <p>Description: The literal HAS_HEADERS instructs the Excel pre-parser to use the first column as the tag names for the resulting XML document.</p> <p>Example: excel (HAS_HEADERS)</p>
	NO_HEADERS	<p>Type/Value: string literal</p> <p>Description: The literal NO_HEADERS instructs the Excel pre-parser to generate column number tag names (for example, <col1>, <col2>) for the resulting XML document.</p> <p>Example: excel (NO_HEADERS)</p>
mfl	MFL routine name	<p>Type/Value: string</p> <p>Description: The name of the MFL routine. Do not enter an .mfl extension for a Message Format Language (MFL) routine; these files are not stored with an extension.</p> <p>Example: mfl (mfl_name)</p>

Table 5-1 Non-XML Pre-Parse Transform Parameters

Transform Type	Parameter	Value/Description/Example
transform	transform file name	Type/Value: string Description: The name of the transform file including the .xch extension. Example: <code>transform(CSVtoXML.xch)</code>

XML to XML Transformations

An event adapter or service adapter may need to convert data from one type of XML document to another. You can choose between two types of transforms to accomplish this conversion:

- *XCH*, a general conversion system which uses dictionaries (implemented as .dic files) and transformation templates (implemented as .xch files).
- *XSLT*, a language for transforming XML documents into other XML documents. It is part of XSL, the XML stylesheet language.

You can invoke either or both transforms during this phase. If you specify both, XSLT occurs after XCH.

For example, in the following figure, the `Order_excel` event specifies an XCH transform that uses the `Order2OrderIn.xch` file and an XSLT transform that uses the `Order2HTML.xsl` stylesheet.

Figure 5-2 Specifying XCH and XSLT Transforms in Add Event Window

Add Event - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://bosrep1:7001/BEA_MQSERIES_1_0_Web/display.jsp Go

Add Event

Application View Console WebLogic Console Glossary Logout

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

Configure Connection
Administration
Add Service
Add Event
Deploy Application View

Unique Event Name*:

☒ **MQEvent**

non-XML Preparse	<input type="text" value="excel(HAS_HEADERS)"/>
XCH Transform	<input type="text" value="Order2OrderIn.xch"/>
XSLT Transform	<input type="text" value="OrderIn2HTML.xsl"/>
Queue Manager*	<input type="text" value="QMBEA"/>
Queue Name*	<input type="text" value="TEST.excelQ"/>
Retry Duration	<input type="text"/>
Retry Interval	<input type="text"/>
Recycle Interval	<input type="text"/>
MQ Client Host	<input type="text"/>
MQ Client Port	<input type="text"/>
MQ Client Channel	<input type="text"/>
Execution Time Limit	<input type="text"/>
Polling Interval	<input type="text"/>

schema:

Done Local intranet

The parameters for each type of transform are shown in the following table.

Table 5-2 XML to XML Transform Parameters

Transform Type	Parameter	Value/Description/Example
XCH Transform	transformation template filename	Type/Value: string Description: The name of the transformation template file including the <code>.xch</code> extension. Example: <code>OrderIn2Out.xch</code>
XSLT Transform	XSLT stylesheet filename	Type/Value: string Description: The name of the transformation template file including the <code>.xch</code> extension. Example: <code>OrderInHTML.xml</code>

XML to Non-XML Transformations

A service adapter that deals with non-XML data sources must be able to convert XML to those non-XML formats for processing by the Enterprise Information System. This conversion entails “pre-emitting” the data into the non-XML format, which is then emitted to the Enterprise Information System.

You can choose between two types of transforms—also known as pre-emitters—to accomplish this conversion:

- *MFL (Message Format Language)*, which uses BEA routines.
- *XCH*, a general conversion system, which uses dictionaries (implemented as `.dic` files) and transformation templates (implemented using `.xch` files).

You can invoke only one transform during this phase—that is, a service adapter can invoke only one kind of XML to non-XML conversion.

For example, in the following figure, the `OrderIn` event specifies XCH and XSLT transforms for the XML to XML phase, as well as an XCH transform for the XML to non-XML pre-emit phase.

Figure 5-3 Specifying XML to Non-XML Transform in Add Service Window

Add Service - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://bosrep1:7001/BEA_MQSERIES_1_0_Web/display.jsp Go

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

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

Unique Service Name: *

MQService

XCH_Transform	<input type="text"/>
XSLT_Transform	<input type="text" value="OrderOut.xsl"/>
non-XML_Pre-Emit	<input type="text" value="OrderOut2csv.xch"/>
Pre-Emit_Engine	<input type="text" value="xch"/>
Queue_Manager*	<input type="text" value="QMBEA"/>
Queue_Name*	<input type="text" value="TEST.oQ"/>
Correlation_Id	<input type="text"/>
Retry_Duration	<input type="text"/>
Retry_Interval	<input type="text"/>
Recycle_Interval	<input type="text"/>
MQ_Client_Host	<input type="text"/>
MQ_Client_Port	<input type="text"/>
MQ_Client_Channel	<input type="text"/>
Execution_Time_Limit	<input type="text"/>
Polling_Interval	<input type="text"/>

schema:

settings

Trace_on/off ☐

Done Local intranet

When you specify XML to non-XML transformation in the Add Service window, specify a transform type by selecting a pre-emit engine and entering the pre-emit parameter.

The parameters for each type of transform (that is, for each pre-emitter) are shown in the following table.

Table 5-3 Non-XML Pre-Emit Transform Parameters

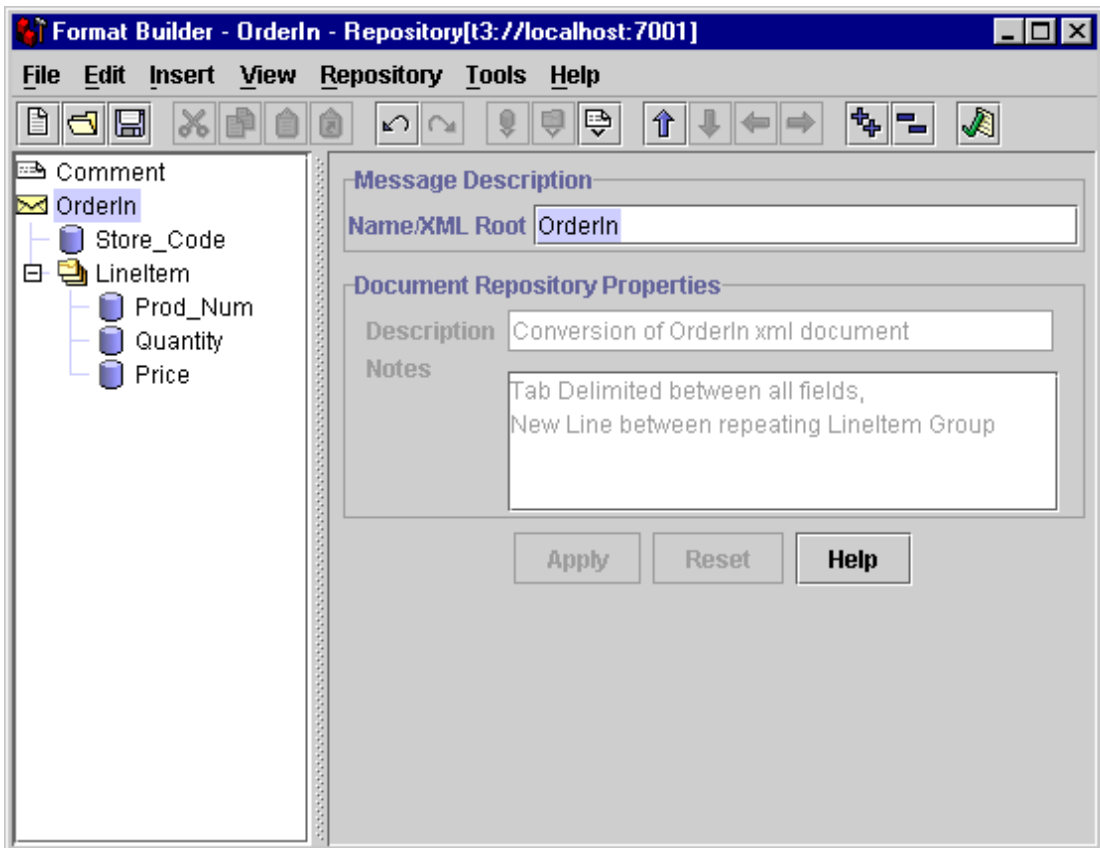
Transform Type	Parameter	Value/Description/Example
mfl	MFL routine name	Type/Value: string Description: The name of the MFL routine. Do not enter an <code>.mfl</code> extension for a Message Format Language (MFL) routine; these files are not stored with an extension. Example: <code>my.mfl</code>
transform	transform file name	Type/Value: string Description: The name of the transform file including the <code>.xch</code> extension. Example: <code>CSVtoXML.xch</code>

Creating MFL Transformations

To create a basic Message Format Language (MFL) transformation using the WebLogic Integration Format Builder:

1. Construct the OrderIn message format in WebLogic Integration Format Builder.

Figure 5-4 WebLogic Format Builder Main Window



2. Save the message format to the repository by choosing Store from the Repository menu and then selecting the desired folder in the repository.

3. Click Store.

Figure 5-5 Store Document Window

Store Document

Current Folder: WLAI.Namespace.Root.MQSeries

Name: OrderIn

File Type: MFL Files

Description: Conversion of OrderIn xml document

Notes: Tab Delimited between all fields,
New Line between repeating LineItem Group

Store

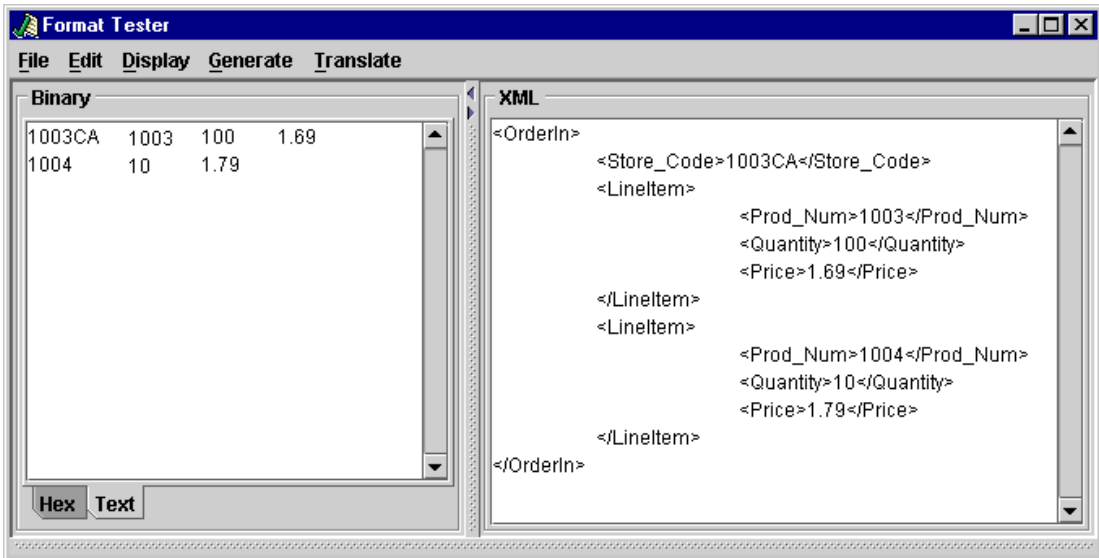
Cancel

There is no need to enter file extensions.

4. Use the Format Tester to provide a preliminary confirmation of the transform: in the XML pane, enter your text.

5. Choose Translate→XML to Binary. The binary representation displays in the Binary pane.

Figure 5-6 Format Tester Window



Once the MFL Format Definition has been stored in the repository, it may be referenced by the BEA WebLogic Adapter for MQSeries.

To test an MFL transformation using the WebLogic Application View Console, see [“Testing MFL Transformations” on page 5-13](#).

Testing MFL Transformations

This procedure describes how to test an MFL transformation using the WebLogic Application View Console. It illustrates the general procedure using a transformation in a service.

1. In the Application View Console, select Add Service from the service adapter application view.

The Add Service window opens.

Figure 5-7 Add Service Window

Add Service

Application View Console 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: *

MQService

XCH Transform	<input type="text"/>
XSLT Transform	<input type="text"/>
non-XML Pre-Emit	<input type="text" value="OrderIn"/>
Pre-Emit Engine	<input type="text" value="mfl"/>
Queue Manager*	<input type="text" value="QMBEA"/>
Queue Name*	<input type="text" value="TEST.oQ"/>
Correlation Id	<input type="text"/>
Retry Duration	<input type="text"/>
Retry Interval	<input type="text"/>
Recycle Interval	<input type="text"/>
MQ Client Host	<input type="text"/>
MQ Client Port	<input type="text"/>
MQ Client Channel	<input type="text"/>
Execution Time Limit	<input type="text"/>
Polling Interval	<input type="text"/>

schema:

Done Local intranet

2. Update the necessary service parameters in the Add Service window.
3. Click Add to save your parameters.
The Administration window opens.
4. Click Continue.
The Deploy Application View window opens.
5. Click Deploy to deploy the new service.
The Summary window opens.
6. Click Test for the new service.
The Test Service window opens.

Figure 5-8 Test Service Window



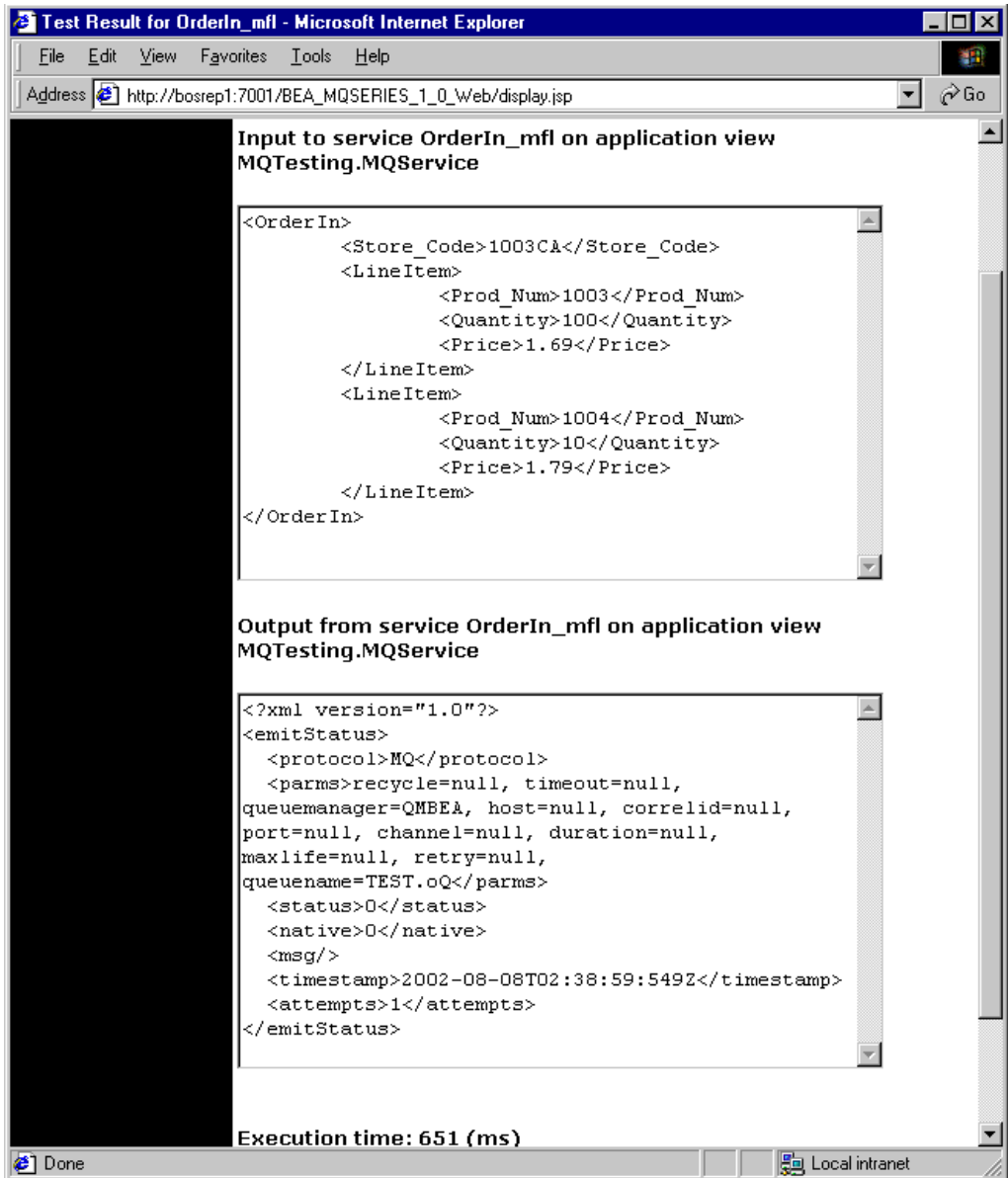
7. Enter a sample XML document into the Test Service window, either by typing it or by copying and pasting it into the window.

8. Click Test to send the XML document to be transformed by the adapter into its binary equivalent, as shown in [Figure 5-6, “Format Tester Window,” on page 5-13](#) under “[Creating MFL Transformations.](#)” The Test Result window opens.

The output from service `OrderIn_mfl` on application view `MQTesting`. `MQService` shows the success or failure status of the transformation in the lower pane of the Test Result window.

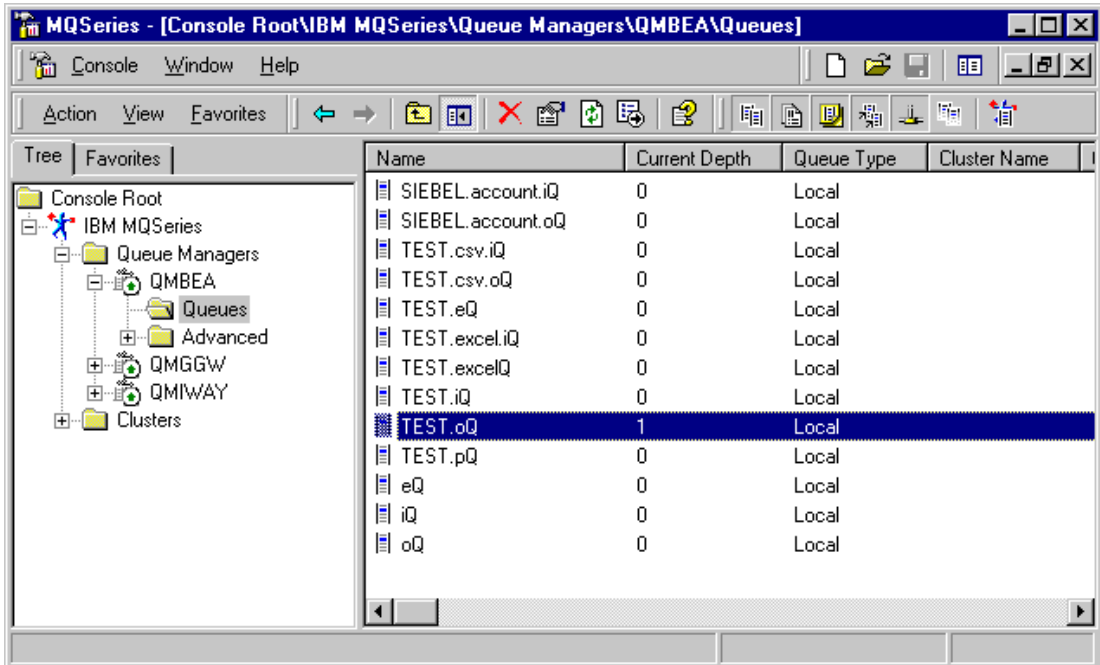
The results of the transformation have now been sent to the output queue.

Figure 5-9 Test Result Window



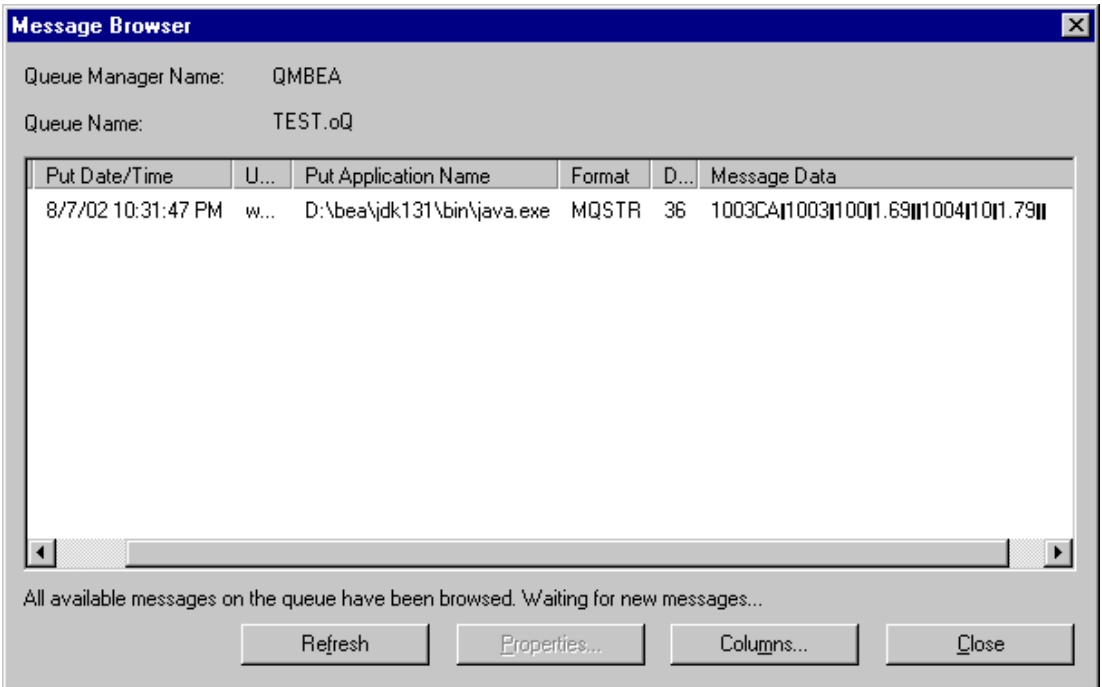
9. In IBM MQSeries Explorer, browse the message queues to view the new message.

Figure 5-10 MQSeries Explorer



10. Right-click the output queue, and choose Browse Messages.... The Message Browser opens.

Figure 5-11 Message Browser Window



The Message Data column displays the transformation's results (including the binary field separators). The results should confirm a successful transformation.

A Error Messages and Troubleshooting

This section lists error messages that you may encounter while using the BEA WebLogic Adapter for MQSeries. It describes what may have generated each error message, and what you can do to resolve the problem.

W.SMQ.1 Failed MQ operation Completion Code 2, Reason 2085

Description	This MQSeries error occurs when the adapter is unable to locate the queue that was specified during configuration of the MQ Listener. Either the queue has not been defined or it has been misspelled. Note that the BEA WebLogic Adapter for MQSeries is case sensitive.
Action	Verify that the queue defined actually exists. Check the spelling and case of the queue name. Use MQSeries Explorer (or the <code>DISPLAY QL</code> command) to verify that the queue is defined.

**W.SMQ.1: {4.11} setupQM: error received java.lang.UnsatisfiedLinkError:
no mqjbnd02 in java.library.path**

Description	This MQSeries error occurs when the system path was not updated to include the directory containing <code>mqjdbnd02.dll</code> .
Action	Refer to the <i>BEA WebLogic Adapter for MQSeries Installation and Configuration Guide</i> and ensure that <code>%MQJAVA%\lib</code> is added to the system path. You must stop and restart WebLogic Integration for this to take effect.

**Cannot load MQ Series. Probable classpath error. java.lang.NoClassDefFoundError:
com/ibm/mq/MQException**

Description	This adapter error occurs when the system path was not updated to include the directory containing mqjdbnd02.dll.
Action	Refer to the <i>BEA WebLogic Adapter for MQSeries Installation and Configuration Guide</i> and ensure that %MQJAVA%\lib\com.ibm.mq.jar and com.ibm.mqbind.jar are added to the system path. You must stop and restart WebLogic Integration for this to take effect.

**Cannot load MQ Series. Probable classpath error. Java.lang.NoClassDefFoundError:
javax/resource/ResourceException**

Description	Please refer to http://www7b.boulder.ibm.com/vajdoc/vahwebx.exe/en_US/vj32/Extract/0/j2sdk/ref/java.lang.NoClassDefFoundError_dsc.html for a detailed explanation of this adapter error.
Action	Refer to the <i>BEA WebLogic Adapter for MQSeries Installation and Configuration Guide</i> and ensure that connector.jar is added to CLASSPATH. You must stop and restart WebLogic Integration for this to take effect.
