



BEA WebLogic Adapter for ISO15022™

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 ISO15022 User Guide

Part Number	Date
N/A	November 2002

Table of Contents

About This Document

What You Need to Know	viii
Related Information.....	viii
Contact Us!	ix
Documentation Conventions	x

1. Introducing the BEA WebLogic Adapter for ISO15022

Components of the BEA WebLogic Adapter for ISO15022.....	1-3
How the BEA WebLogic Adapter for ISO15022 Works.....	1-4
Validation	1-5
Inbound.....	1-6
Outbound.....	1-6
Configuring and Validating ISO15022 Documents	1-6

2. Creating and Configuring an Event Adapter

Creating an Application View Folder.....	2-2
Creating an Event Adapter Application View.....	2-6
Configuring an Event Adapter Application View	2-10
Testing Event Adapter Application Views Using the Application View Console..	2-22
Testing Event Adapter Application Views Using WebLogic Integration Studio ...	2-27

3. Creating and Configuring a Service Adapter

How to Create a Service Adapter Application View.....	3-2
How to Configure a Service Adapter Application View.....	3-6
Testing the BEA Service Adapter for ISO15022	3-17

4. Transforming Document Formats

Kinds of Transformation	4-2
SWIFT to XML Transformations	4-2
XML to XML Transformations	4-3
XML to SWIFT Transformations	4-4

5. Acknowledgement Handling

Acknowledgement Processing	5-2
Validation	5-2
Document Tree	5-3
Acknowledgement Agent	5-3
Documents, Validation, and Acknowledgement	5-3
Acknowledgement Agent	5-4
Acknowledgement Message Handling	5-6
Creating an Acknowledgement Event	5-7
Testing Acknowledgement Message Handling	5-14

A. BEA WebLogic Adapter for ISO15022 Rules System

Rules File	A-2
General Rule Set	A-4
isN	A-5
isR	A-5
isDate	A-5
isTime	A-6
SWIFT Specific Rule Set	A-6
isValidReference	A-7
isValidISIN	A-7
isNotPresent	A-8
isValidMultiLine	A-8
isSWIFTReal	A-9
isSWIFTDate	A-9
IsValidSWIFTString	A-10
Hexadecimal Representation of SWIFT Character Set	A-12
isSWIFTTime	A-13
isValidMessageType	A-13

checkValue	A-14
checkCD	A-15
checkRepetitive	A-17
checkNodes	A-18
checkChildSequence	A-18
checkAddition	A-19
checkRelation	A-20
checkSegment	A-20
Writing Rules in Java	A-21
Writing Rule Search Routines in Java	A-24

B. Linking the BEA WebLogic Adapter for ISO15022 to a SWIFT Network

Batch File Transfer – FILE and FTP	B-3
Application Server – CAS MF	B-3
Interactive – MQ Series	B-4



About This Document

The *BEA WebLogic Adapter for ISO15022 User Guide* is organized as follows:

- [Chapter 1, “Introducing the BEA WebLogic Adapter for ISO15022,”](#) introduces the BEA WebLogic Adapter for ISO15022, describes its features, and gives an overview of how it works.
- [Chapter 2, “Creating and Configuring an Event Adapter,”](#) describes how metadata is used and how application views are created.
- [Chapter 3, “Creating and Configuring a Service Adapter,”](#) describes how to add services and events to application views.
- [Chapter 4, “Transforming Document Formats,”](#) describes how events are incorporated into workflow design.
- [Chapter 5, “Acknowledgement Handling,”](#) describes the Acknowledgement process.
- [Appendix A, “BEA WebLogic Adapter for ISO15022 Rules System,”](#) describes how documents are to be validated against sets of rules.
- [Appendix B, “Linking the BEA WebLogic Adapter for ISO15022 to a SWIFT Network,”](#) describes the connecting of business applications to SWIFTAlliance.

What You Need to Know

This document is written for system integrators who develop client interfaces between ISO15022 and other applications. It describes how to use the BEA WebLogic Adapter for ISO15022 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 ISO15022 Installation and Configuration Guide*
- *BEA WebLogic Adapter for ISO15022 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 ISO15022 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 ISO15022 documentation.

In your e-mail message, please indicate which version of the BEA WebLogic Adapter for ISO15022 documentation you are using.

If you have any questions about this version of the BEA WebLogic Adapter for ISO15022, or if you have problems using the BEA WebLogic Adapter for ISO15022, 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> #include <iostream.h> void main () the pointer psz chmod u+w * \tux\data\ap .doc tux.doc BITMAP float
monospace boldface text	Identifies significant words in code. <i>Example:</i> void commit ()
<i>monospace italic text</i>	Identifies variables in code. <i>Example:</i> String <i>expr</i>
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. <i>Examples:</i> LPT1 SIGNON OR

Convention	Item
{ }	Indicates 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 ISO15022

This section introduces the BEA WebLogic Adapter for ISO15022, describes its features, and gives an overview of how it works. It includes the following topics:

- [Components of the BEA WebLogic Adapter for ISO15022](#)
- [How the BEA WebLogic Adapter for ISO15022 Works](#)
- [Configuring and Validating ISO15022 Documents](#)

SWIFT networks carry messages between financial institutions and must originate or arrive at financial institutions in a format described by a SWIFT message standard. In 2002, the SWIFT standards body moved from ISO Standard 7775 to ISO Standard 15022 and SWIFT documents began conforming to this standard.

The BEA WebLogic Adapter for ISO15022 transforms documents into XML format and converts between message formats in 7775 format into 15022 format. The adapter provides bi-directional transformation to ensure integration to and from 15022 compliant networks and applications.

With the XML document in the 15022 format, the information can be integrated into back or front office 7775 standard systems using WebLogic Integration and all of the BEA application and data adapters that are available from the adapter suite of products.

J2EE™ standard interfaces and protocols such as JCA™, JDBC™, and JMS™ are also supported with the BEA WebLogic Adapter for ISO15022. The same adapters can be used to obtain information necessary to populate SWIFT messages. For example, you can use the BEA WebLogic Adapter for RDBMS updates in an RDBMS to trigger a SQL query that returns an XML formatted answer set that can be mapped to a SWIFT message.

Data dictionaries describe the SWIFT message format and are used to enable the mapping of XML documents to SWIFT form and SWIFT messages to XML. After structural integrity has been verified during the transformation stage, the BEA WebLogic Adapter for ISO15022 performs validation, using a set of rules defined in an XML formatted rules file. Where applicable, acknowledgment documents are returned to the sending application, but only if the incoming document is structurally valid. If the content validation fails, an error code is returned in the acknowledgement document, if one is expected.

You can use the adapter to exchange SWIFT formatted documents over any one of the supported transport protocols with WebLogic Integration to provide a tightly integrated application infrastructure. The multiprotocol support of the adapter allows for the greatest flexibility in application integration from Web-based form submission to guaranteed delivery using WebSphere MQSeries.

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 ISO15022 provides:

- *Multi-protocol support* for integration with any form of application system or standardized message handling system.
- *Message transfer* between SWIFT message handling systems and WebLogic Integration.
- *Service and event adapter integration operation* providing end-to-end business process management using SWIFT formatted messages and XML schema defined business processes.

- *Support for custom and standard SWIFT message formats* with automatic generation of transforms into a common XML business process environment.
- SWIFT format conversion paths from ISO 7775 to 15022 and from ISO 15022 to 7775.

The adapter provides pre-packaged support for 15022 standard documents, but does not provide out-of-the-box the ability to customize those formats. Please contact BEA professional services if you need to customize these formats.

Components of the BEA WebLogic Adapter for ISO15022

The BEA WebLogic Adapter for ISO15022 is packaged with the components required to integrate SWIFT message information into an existing ISO 7775 systems environment. The following components are supplied with the adapter:

- *WebLogic Integration Application Views*. This is the backbone of the BEA WebLogic Adapter for ISO15022. WebLogic Integration is the host for the adapter and can be used to host a multitude of BEA adapters. This architecture enables easy integration spanning multiple protocols (for example, SWIFT and MQ Series) and back-end systems and/or data sources.
- *BEA WebLogic Adapter for ISO15022 kit*. The BEA WebLogic Adapter for ISO15022 kit is a set of components that enable the integration of SWIFT message formats into an environment. The following list details the components supplied with the kit:
 - *XML schemas (.xsd)*. These describe the SWIFT messages for event and service processing.
 - *Data dictionaries (.dic)*. These describe the SWIFT format messages to the BEA WebLogic Adapter for ISO15022.
 - *Transformation templates (.xch)*. These are the maps that are created using the transformation templates configured to convert SWIFT to XML and XML to SWIFT and between ISO 7775 and ISO 15022 standard message formats.

- *Rules files (.xml)*. Rules files are XML documents that are used to apply business rules to the SWIFT XML file (either post or prior to conversion). The rules files are built to the SWIFT standards specification and are customizable to apply customer and/or partner specific rules.
- *Code sets (.txt)*. Currency and country (ISO standard) code set tables are provided for validation using the rules engine.

How the BEA WebLogic Adapter for ISO15022 Works

The BEA WebLogic Adapter for ISO15022 provides transport protocol, transformation, and document validation support. It can receive and emit SWIFT formatted documents over any supported protocol, transform the documents into XML standard format, and validate the document against a suite of SWIFT rules.

When the BEA WebLogic Adapter for ISO15022 receives a document, it processes the document in a number of ways. The BEA WebLogic Adapter for ISO15022 supports bi-directional transformation using XML as the intermediate format across a number of transport protocols. The adapter is supplied with templates to convert SWIFT message types to XML. Schemas are supplied to SWIFT format.

The SWIFT documents are described using data dictionaries supplied with the adapter. The data dictionaries conform to the SWIFT standards and are updated with each year's amendments, and the format for describing data elements conform to the standards in the books. These dictionaries are in XML format and can be edited to tailor messages to individual bank and/or market standards.

Receipt of SWIFT messages has been implemented as an event adapter within WebLogic Integration. Emission of SWIFT messages has been implemented as a service adapter. These adapters provide a number of processing options that you can configure with the design-time WebLogic Integration Application View.

- *Transformation services.* XML is quickly becoming the standard for exchanging information between applications; it is invaluable in integrating disparate applications. The BEA WebLogic Adapter for SWIFT automatically recognizes the SWIFT message types and converts them into the XML equivalent. The adapter provides the full suite of message formats and their associated XML Schema Definitions (.xsd files).
- *Document validation rules.* During the analysis of an incoming SWIFT message, the system validates the message against the associated SWIFT rule. 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.
- *Message acknowledgment.* Message receipt and message validation may be configured to generate message validation. A validation response may be emitted on any of the supported transports.
- *Protocol emitting.* The service adapter supports emitting SWIFT messages to any of the supported transport protocols.

Validation

The SWIFT messages are validated in two ways. The first way is by using the dictionaries. The dictionaries are used to parse the document and validate the structure of the message type and tag structure. The second way is by the rules engine, which provides content (domain and network) validation.

The SWIFT document is validated by use of a rule file in XML format. This file is an XML document that applies pre-built rules based on the XML tag. These are customizable, and users can write their own rules to apply extra business logic.

Depending on the direction (XML to SWIFT or SWIFT to XML), the content validation occurs before or after structural validation. The following section explains how transformation and validation works for both inbound (receiving a SWIFT message) and outbound (creating a SWIFT message) processes.

Inbound

A configured listener picks up an inbound SWIFT message. The first step is the pre-parse stage. Because the iXTE maps input and output using XML format, the document must be converted to XML. This is where the transformation templates are called to convert SWIFT format to XML.

After the content is in XML format, the content of the SWIFT document can be validated based on supplied rules files. BEA supplies pre-built rules that apply validation rules to tags or groups of tags in the XML document, specific to the SWIFT green books.

Outbound

In the outbound process used by the BEA WebLogic Adapter for ISO15022, an XML document can be transformed to XML or to SWIFT ISO 15022 standard. This can be from a custom XML or from a 7775 standard document. The transformations from 7775 to 15022 and back are supplied.

A document can be received, in XML format, and have business logic applied. The rules engine then validates the document and transforms it into a SWIFT document at the pre-emit stage in the process.

Configuring and Validating ISO15022 Documents

The BEA WebLogic Adapter for ISO15022 comes with a WebLogic JSP-based console that allows authorized users to manage the engine. Listeners, validation rules, and other configuration tasks can be performed here.

Validation occurs against the XML representation of the SWIFT documents. Conversion to XML occurs first. At this stage, the SWIFT dictionaries are used, and structural validation is applied.

Further network and content validation of the SWIFT documents using the BEA WebLogic Adapter for ISO15022 is performed by a Rules Class, applying rules defined in the rules files. These rules must be created, assigned to the adapter (that is, an alias is created), and applied to the document. BEA supplies standard rules files that validate a document based on the SWIFT 2001 supplied standards.

These files are pre-configured and known to the adapter and also have been related to the documents to be validated. No configuration is required. If customized rules files are used and applied to custom built transformation templates, see [Appendix A, “BEA WebLogic Adapter for ISO15022 Rules System.”](#) This section documents the standard rules and SWIFT-specific rules supplied with the BEA WebLogic Adapter for ISO15022.

The Validation phase may or may not produce errors related to elements in the document tree. After validation, the acknowledgement agent processing begins. This phase allows for custom behavior based on the results of the document and its validation results. For ISO15022 SWIFT messages, the default acknowledgement agent is the XDSWIFTACKAgent. The acknowledgement process is described in detail in [Chapter 5, “Acknowledgement Handling.”](#)

2 Creating and Configuring an Event Adapter

An event adapter is the inbound interface from an external protocol. For SWIFT ISO 15022, this may be an MQSeries resource manager (that is, from a Queue Manager and its associated queues), FILE, FTP, TCP, and so forth. This section describes how metadata is used and how application views are created. It 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 the 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:

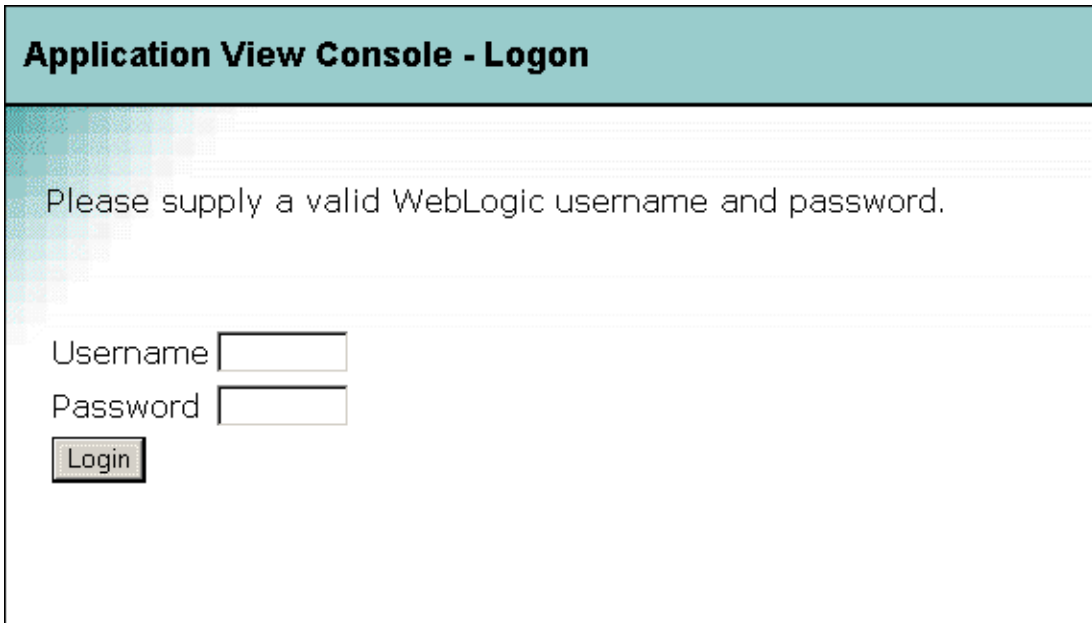
1. Log on to the WebLogic Integration Application View Console at the following location:

`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. If prompted, enter a user name and password, as shown in the following figure.

Figure 2-1 Application View Logon



Application View Console - Logon

Please supply a valid WebLogic username and password.

Username

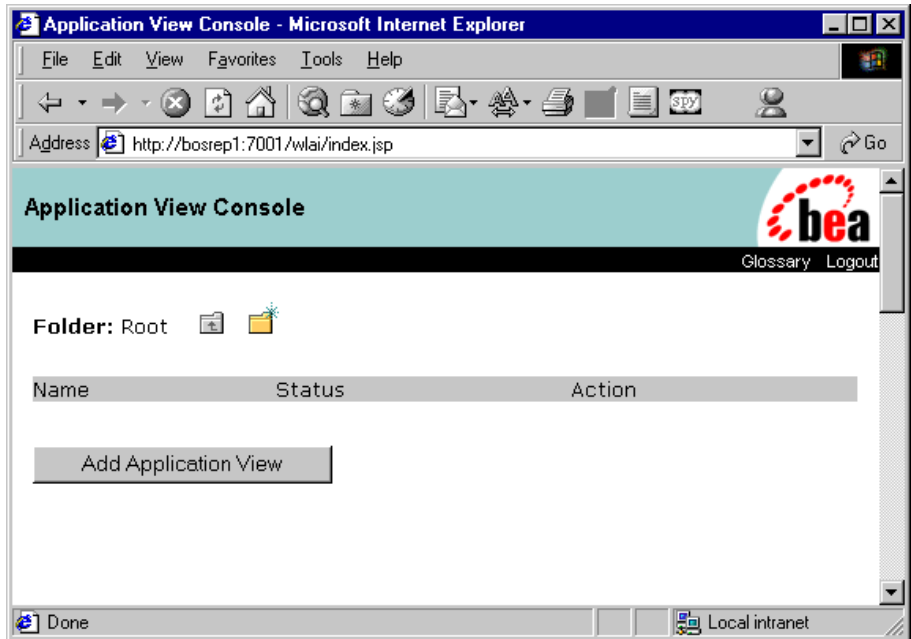
Password

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 ISO15022 Installation and Configuration Guide*.

3. Click Login.

The Application View Console opens.

Figure 2-2 Application View Console Window



4. Click the new folder icon.
The Add Folder page opens.

Figure 2-3 Application View Console: Add Folder



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

You have created the application view folder.

Creating an Event Adapter Application View

To create an event adapter application view:

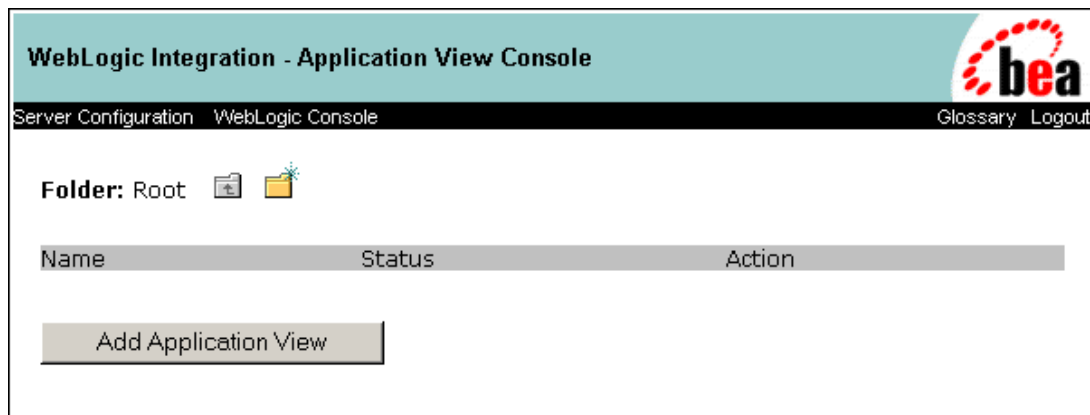
1. Open the Application View Console, which is found at the following location:

`http://host:port/wlai`

Here, *host* is the TCP/IP address or DNS name where WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The default port at the time of installation is 7001.

The Application View Console opens.

Figure 2-4 Application View Console



2. Click Add Application View to create an application view for the adapter. The Define New Application View dialog box opens. An application view enables a set of business processes for this adapter's target EIS application. For more information, 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

The Define New Application View page opens.

Figure 2-5 Application View Console: Define New Application View

The screenshot shows a web browser window titled "Application View Console - Microsoft Internet Explorer". The address bar shows the URL: `http://bosrep1:7001/wlai/display.jsp?content=defappvw&namespace=ISO`. The page has a header with the BEA logo and links for "Glossary" and "Logout". The main content area is titled "Define New Application View" and contains the following form fields:

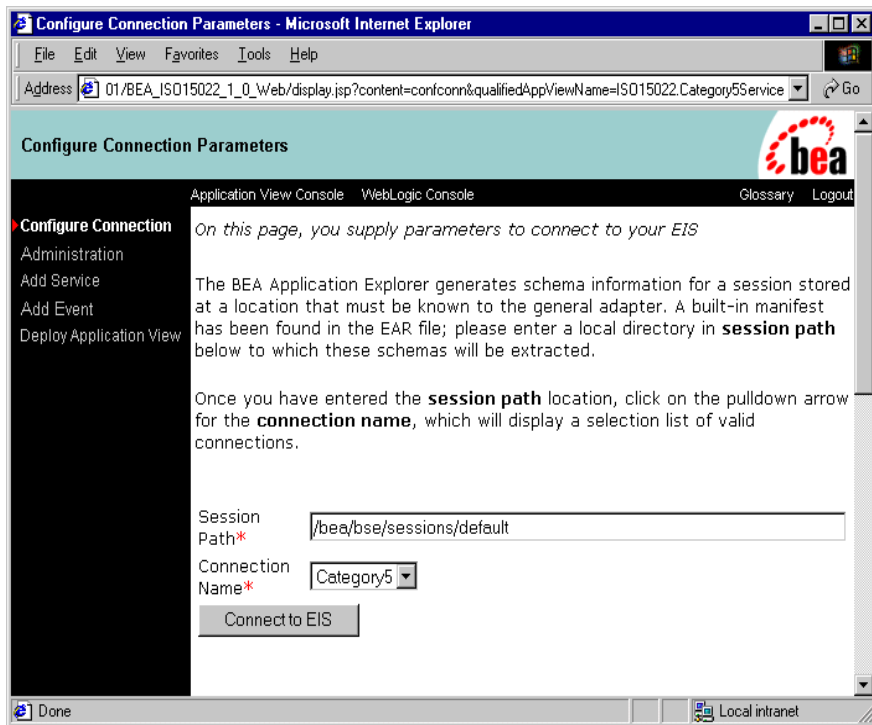
- Folder:** ISO15022
- Application View Name:** * Category5Service
- Description:** BEA Adapter for ISO15022 - Service Category 5 Services
- Associated Adapter:** BEA_ISO15022_1_0

At the bottom of the form are "OK" and "Cancel" buttons. The browser's status bar shows "Done" and "Local intranet".

3. Type a name and description for the application view.
4. Select BEA_ISO15022_1_0 from the Associated Adapter drop-down list:
5. Click OK.

The Configure Connection Parameters page opens.

Figure 2-6 Application View Console: Configure Connection Parameters



6. Type the name of the BEA WebLogic Adapter for ISO15022 session base directory in the Session path box.

This directory holds your ISO15022 SWIFT schema information and contains the subdirectory `ISO15022/YourConnectionName`.

For example, the session base directory might be `/bea/bse/sessions/default`, with the schema repository—containing a repository manifest and schemas—residing in `/bea/bse/sessions/default/ISO15022/Category5`.

7. From the Connection name drop-down list, select the session name (also known as the connection name).

- Click Connect to EIS.

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

The Application View Administration page opens.

Figure 2-7 Application View Console: Application View Administration



- 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 2-10, 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 service should be added for each event 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 `//appserver-host:port/wlai`.
2. Select the folder in which this application view resides and then select the application view.
3. In the Administration page of the WebLogic Integration Application View Console, select Add Event.

The Add Event page opens.

Figure 2-8 Application View Console: Add Event - MQSeries

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: *

Select: MQ_ISO15022

XSLT Transform	
Queue Manager*	QMBEA
Queue Name*	ISO15022.iQ
Recycle Interval	
MQ Client Host	
MQ Client Port	
MQ Client Channel	
Execution Time Limit	
Polling Interval	
ackagent	XDSWIFTACKAgent

schema: 571

settings

Trace on/off	<input type="checkbox"/>
Logging on/off	<input type="checkbox"/>
Deep Debug on/off	<input type="checkbox"/>
Maximum Log Size (kb)	500
root to transform template directory	L:/SCHEMAS/ISO15022/category6/te
root to XML Style sheet directory	L:/SCHEMAS/ISO15022/category6/te

Add

The BEA WebLogic Adapter for ISO15022 supports multiple protocols for receiving its SWIFT messages.

- In the Select drop-down list box, select the desired transport protocol.

The screen displays the associated parameters for the chosen protocol.

2 Creating and Configuring an Event Adapter

The properties in the previous window correspond to the MQSeries communication and transformation settings that the event adapter uses. The schema drop-down list corresponds to the list of events in the schema repository.

The following table describes the communication settings for the MQSeries transport protocol.

Table 2-1 Event Properties - MQSeries

Property	Description	Type	Sample Value	Element
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 4, “Transforming Document Formats.”	string		<in_xslt>
Queue Manager* (Required)	Name of the MQSeries Queue Manager to be used.	string	OMBEA	<manager>
Queue Name	Queue on which request documents are received.	string	ISO15022.iO	<queue>
Recycle Interval	Interval between retrying successful requests.	duration		<recycle>
MQ Client Host	For MQ Client only. Host on which MQ Server is located.	string		<host>
MQ Client Port	For MQ Client only. Port number to connect to an MQ Server.	integer		<port>
MQ Client Channel	For MQ Client only. Channel between an MQ Client and MQ Server.	string		<channel>
Execution Time Limit	The length of time in seconds before execution is terminated.	duration		<maxlife>

Table 2-1 Event Properties - MQSeries (Continued)

Property	Description	Type	Sample Value	Element
Polling Interval	The time in milliseconds.	duration		<timeout>
ackagent	The agent responsible for processing acknowledgement or non-acknowledgement of ISO SWIFT documents. The default SWIFT acknowledgement agent will generate an empty acknowledgement document or will list all failed validation rules in the non-acknowledgement document.	string		<ackagent>

For the File adapter, the following screen shows the applicable parameters:

Figure 2-9 Application View Console: Add Event - File

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:*

Select: File_ISO15022

XSLT Transform	<input type="text"/>
Location*	<input type="text" value="d:/input/directory/pattern*"/>
File Suffix*	<input type="text" value="swift"/>
Polling Interval	<input type="text" value="1"/>
Sort	<input type="checkbox"/>
Scan sub-directories	<input type="checkbox"/>
File-read limit (per scan)	<input type="text"/>
ackagent	<input type="text" value="XDSWIFTACKAgent"/>
encoding	<input type="text" value="ISO-8859-1"/>

schema: 571

settings

Trace on/off	<input type="checkbox"/>
Logging on/off	<input type="checkbox"/>
Deep Debug on/off	<input type="checkbox"/>
Maximum Log Size (kb)	<input type="text" value="500"/>
root to transform template directory	<input type="text" value="L:/SCHEMAS/ISO15022/category6/te"/>
root to XML Style sheet directory	<input type="text" value="L:/SCHEMAS/ISO15022/category6/te"/>

Add

The following table describes the properties for File:

Table 2-2 Event Properties - File

Setting	Meaning/Properties
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 4, “Transforming Document Formats.”
Location* (*Required)	Type/Value: String Description: The location and name of the file pattern for the SWIFT documents.
File Suffix* (*Required)	Type/Value: String Description: Mandatory suffix of the input files.
Polling Interval	Type/Value: Duration Description: Time in milliseconds between polls.
Sort* (*Required)	Type/Value: Boolean Description: Flag to indicate whether the list of files pulled from the location should be sorted for processing order.
Scan sub-directories	Type/Value: Boolean Description: Flag to indicate whether files should be read from subdirectories.
File Read limit (per scan)	Type/Value: Integer Description: The number of files read per sweep of the File directory location.
ackagent	Type/Value: String Description: The agent responsible for processing acknowledgement or non-acknowledgement of ISO SWIFT documents. The default SWIFT acknowledgement agent will generate an empty acknowledgement document or will list all failed validation rules in the non-acknowledgement document.
encoding	Type/Value: String Description: Document character set.

2 Creating and Configuring an Event Adapter

For the FTP adapter, the following screen shows the applicable parameters:

Figure 2-10 Application View Console: Add Event - FTP

Add Event

Configure Connection

Administration

Add Service

Add Event

Deploy Application View

Application View Console

WebLogic Console

Glossary

Logout

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

Unique Event Name: *

Select: FTP_ISO15022

XSLT Transform	
Host name *	
Host port	
User Id *	
Password	
Location *	
File Suffix	
Polling interval	
ackagent	XDSWIFTACKAgent
Character Set Encoding *	ISO-8859-1

schema: 571

settings

Trace on/off	<input type="checkbox"/>
Logging on/off	<input type="checkbox"/>
Deep Debug on/off	<input type="checkbox"/>
Maximum Log Size (kb)	500
root to transform template directory	L:/SCHEMAS/ISO15022/category6/te
root to XML Style sheet directory	L:/SCHEMAS/ISO15022/category6/te

Add

Table 2-3 Event Properties - FTP

Setting	Meaning/Properties
XSLT Transform	<p>Name of the XSLT stylesheet used by the XSLT transform type in the XML to XML transformation phase.</p> <p>For more information, see Chapter 4, “Transforming Document Formats.”</p>
Host name* (*Required)	<p>Type/Value: String</p> <p>Description: FTP target system.</p>
Host port	<p>Type/Value: Integer</p> <p>Description: Alternate port number.</p>
User ID* (*Required)	<p>Type/Value: String</p> <p>Description: User account ID to use when connecting to protocol host.</p>
Password	<p>Type/Value: String</p> <p>Description: Password for user account to use when connecting to protocol host.</p>
Location* (*Required)	<p>Type/Value: Directory path and file</p> <p>Description: Location on FTP host to retrieve files from. You must append the file suffix (extension) to the file or files specified in the Location field. For example, you can enter a specific file such as <code>/path/to/my/ftp/directory/myfile.xml</code> or a group of files such as <code>/path/to/my/ftp/directory/*.zip</code>.</p>
File Suffix	<p>Type/Value: String</p> <p>Description: This field is no longer used. You must append the file suffix to the file or files specified in the Location field.</p>
Polling Interval	<p>Type/Value: Integer</p> <p>Description: Time in milli-seconds between polls from the FTP server.</p>

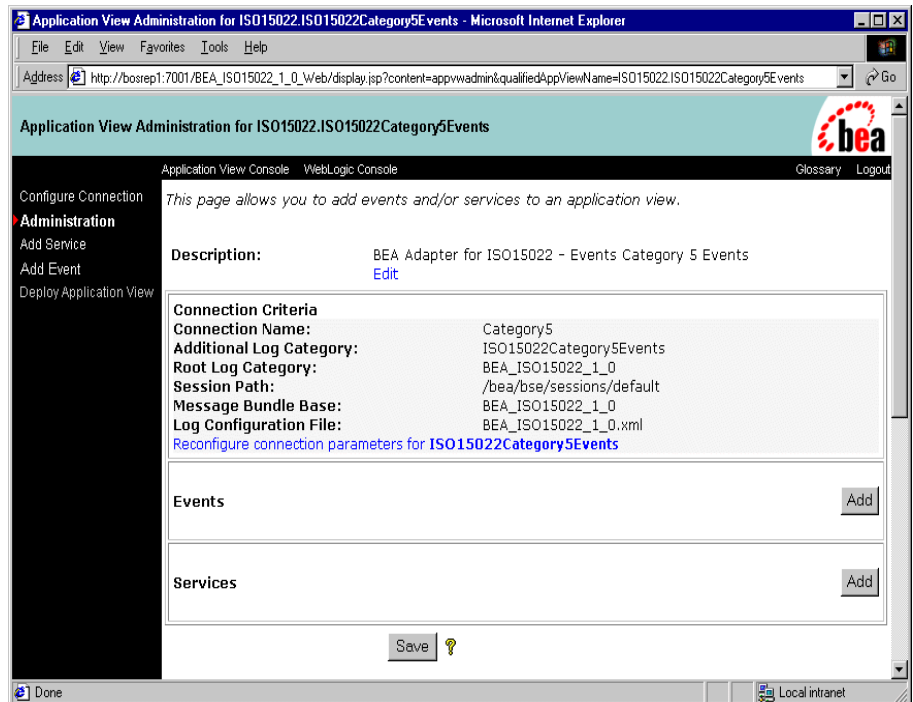
2 Creating and Configuring an Event Adapter

Table 2-3 Event Properties - FTP

Setting	Meaning/Properties
ackagent	Type/Value: String Description: The agent responsible for processing acknowledgement or non-acknowledgement of ISO SWIFT documents. The default SWIFT acknowledgement agent will generate an empty acknowledgement document or will list all failed validation rules in the non-acknowledgement document.
Character set encoding* (*Required)	Type/Value: String Description: Document character set.

5. Click Add. The Application View Administration page opens.

Figure 2-11 Application View Console: Application View Administration



6. Click Continue.

The Deploy Application View page opens.

Figure 2-12 Application View Console: Deploy Application View

The screenshot shows a web browser window titled "Deploy Application View ISO15022.ISO15022Category5Events to Server - Microsoft Internet Explorer". The address bar shows "http://bosrep1:7001/BEA_ISO15022_1_0_Web/display.jsp". The page has a navigation menu on the left with options: "Configure Connection", "Administration", "Add Service", "Add Event", and "Deploy Application View" (which is selected). The main content area is titled "Deploy Application View ISO15022.ISO15022Category5Events to Server" and contains the following sections:

- Required Event Parameters**: A text field for "Event Router URL*" with the value "http://bosrep1:7001/BEA_ISO15022_1_0_EventRouter/Eventf".
- Connection Pool Parameters**: A section with the instruction "Use these parameters to configure the connection pool used by this application view". It includes:
 - "Minimum Pool Size*" with a value of "1".
 - "Maximum Pool Size*" with a value of "10".
 - "Target Fraction of Maximum Pool Size*" with a value of "0.7".
 - "Allow Pool to Shrink?" with a checked checkbox.
- Log Configuration**: A section with the instruction "Set the log verbosity level for this application view." and a dropdown menu set to "Log warnings, errors, and audit messages".
- Configure Security**: A section with a link "Restrict Access to ISO15022Category5Events using J2EE Security".

At the bottom of the form, there are buttons for "Deploy", "Save", and a checkbox for "Deploy persistently?".

7. Modify event parameters, connection pool parameters, log configuration, and security as necessary.
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:

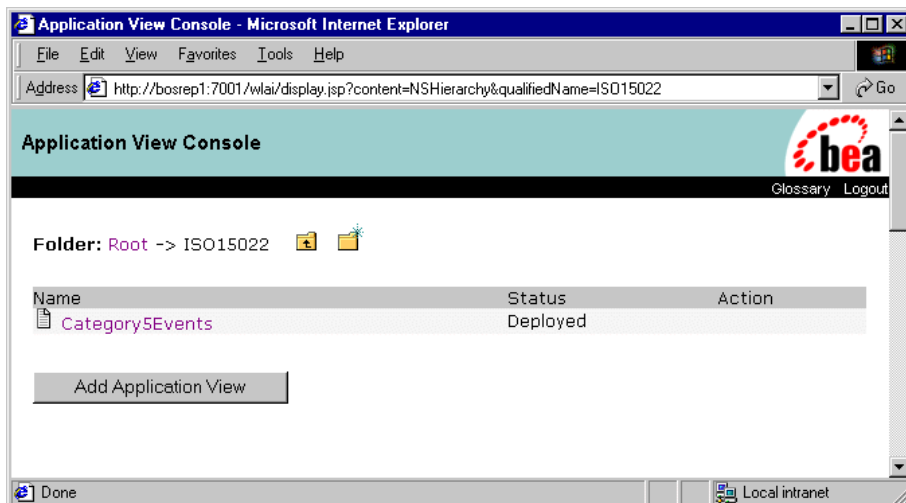
Figure 2-13 WebLogic Server Log

```

WebLogic -start -startweblogic
/Category5/rules/SWIFTM578rules.xml' in validation area
DEBUG 15 Aug 2002 17:14:19,863 BEA_ISO15022_1_0.EventGenerator - DEBUG [dictionary] precompileValidation - '/bea/bse/sessions/default/I
/Category5/rules/SWIFTM580rules.xml' in validation area
DEBUG 15 Aug 2002 17:14:19,893 BEA_ISO15022_1_0.EventGenerator - DEBUG [dictionary] precompileValidation - '/bea/bse/sessions/default/I
/Category5/rules/SWIFTM581rules.xml' in validation area
DEBUG 15 Aug 2002 17:14:19,924 BEA_ISO15022_1_0.EventGenerator - DEBUG [dictionary] precompileValidation - '/bea/bse/sessions/default/I
/Category5/rules/SWIFTM582rules.xml' in validation area
DEBUG 15 Aug 2002 17:14:19,964 BEA_ISO15022_1_0.EventGenerator - DEBUG [dictionary] precompileValidation - '/bea/bse/sessions/default/I
/Category5/rules/SWIFTM583rules.xml' in validation area
DEBUG 15 Aug 2002 17:14:20,034 BEA_ISO15022_1_0.EventGenerator - DEBUG [dictionary] precompileValidation - '/bea/bse/sessions/default/I
/Category5/rules/SWIFTM587rules.xml' in validation area
DEBUG 15 Aug 2002 17:14:20,064 BEA_ISO15022_1_0.EventGenerator - DEBUG [dictionary] precompileValidation - '/bea/bse/sessions/default/I
/Category5/rules/SWIFTM588rules.xml' in validation area
DEBUG 15 Aug 2002 17:14:20,114 BEA_ISO15022_1_0.EventGenerator - DEBUG [dictionary] precompileValidation - '/bea/bse/sessions/default/I
/Category5/rules/SWIFTM589rules.xml' in validation area
INFO 15 Aug 2002 17:14:20,204 BEA_ISO15022_1_0.EventGenerator - INFO [manager] Starting System (version 5.2.1)
INFO 15 Aug 2002 17:14:20,204 BEA_ISO15022_1_0.EventGenerator - INFO [manager] Server code page is Cp1252
DEBUG 15 Aug 2002 17:14:20,204 BEA_ISO15022_1_0.EventGenerator - LOG [MQ_ISO15022] MQ_ISO15022: Protocol= MQ
DEBUG 15 Aug 2002 17:14:20,204 BEA_ISO15022_1_0.EventGenerator - LOG [MQ_ISO15022] MQ_ISO15022: retry = [default] 600
DEBUG 15 Aug 2002 17:14:20,204 BEA_ISO15022_1_0.EventGenerator - LOG [MQ_ISO15022] MQ_ISO15022: precedence = [dict] 2
DEBUG 15 Aug 2002 17:14:20,214 BEA_ISO15022_1_0.EventGenerator - LOG [MQ_ISO15022] MQ_ISO15022: encoding = [dict] ISO-8859-1
DEBUG 15 Aug 2002 17:14:20,214 BEA_ISO15022_1_0.EventGenerator - LOG [MQ_ISO15022] MQ_ISO15022: duration = [default] 86400
DEBUG 15 Aug 2002 17:14:20,214 BEA_ISO15022_1_0.EventGenerator - LOG [MQ_ISO15022] MQ_ISO15022: agent = [dict] XDCopyAgent
DEBUG 15 Aug 2002 17:14:20,214 BEA_ISO15022_1_0.EventGenerator - LOG [MQ_ISO15022] MQ_ISO15022: manager = [dict] QMBEA
DEBUG 15 Aug 2002 17:14:20,214 BEA_ISO15022_1_0.EventGenerator - LOG [MQ_ISO15022] MQ_ISO15022: queue = [dict] ISO15022.iQ
DEBUG 15 Aug 2002 17:14:20,214 BEA_ISO15022_1_0.EventGenerator - LOG [MQ_ISO15022] MQ_ISO15022: count = [dict] 1
DEBUG 15 Aug 2002 17:14:20,214 BEA_ISO15022_1_0.EventGenerator - LOG [MQ_ISO15022] MQ_ISO15022: ackagent = [dict] XDSWIFTACKAgent
DEBUG 15 Aug 2002 17:14:20,224 BEA_ISO15022_1_0.EventGenerator - LOG [MQ_ISO15022] MQ_ISO15022: timeout = [default] 2
MQISO001: Completion Code 1, Reason 1
DEBUG 15 Aug 2002 17:14:22,557 BEA_ISO15022_1_0.EventGenerator - DEBUG [MQ_ISO15022] W.MQ_ISO15022.1: creating preParser 'XDSWIFTprePar
DEBUG 15 Aug 2002 17:14:22,567 BEA_ISO15022_1_0.EventGenerator - DEEP [MQ_ISO15022] W.MQ_ISO15022.1: Agent's parm map: {}
DEBUG 15 Aug 2002 17:14:22,577 BEA_ISO15022_1_0.EventGenerator - DEBUG [MQ_ISO15022] W.MQ_ISO15022.1: adding preParser 'XDSWIFTpreParse
index 0
DEBUG 15 Aug 2002 17:14:22,577 BEA_ISO15022_1_0.EventGenerator - starting W.MQ_ISO15022.1
  
```

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

Figure 2-14 Application View Console: New Application View



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 the Application View Console”](#) on page 2-22 and [“Testing Event Adapter Application Views Using WebLogic Integration Studio”](#) on page 2-27.

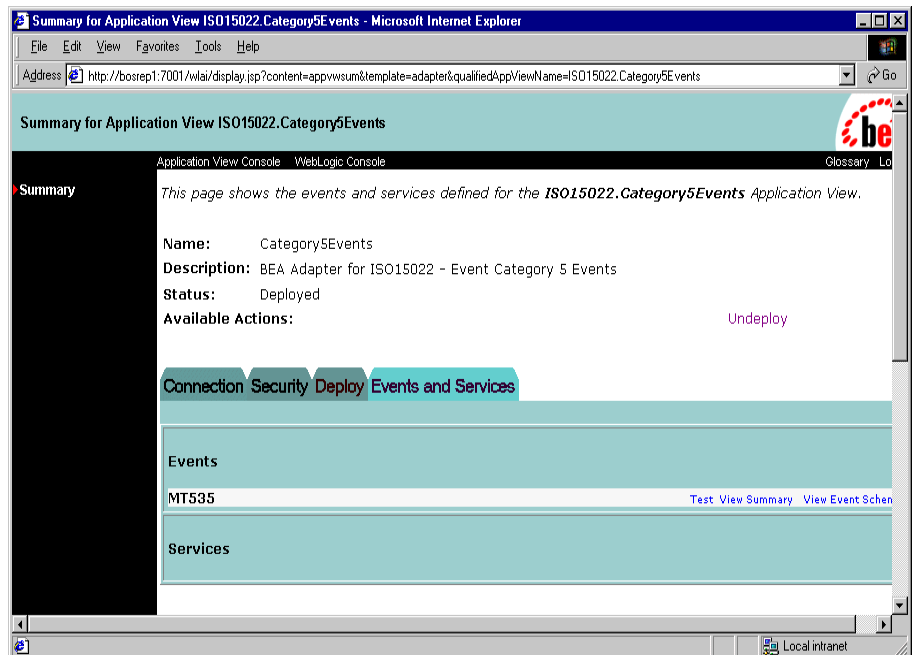
Testing Event Adapter Application Views Using the Application View Console

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 `//appserver-host:port/wlai`.
2. Select the folder in which the application view resides and then select the application view.

The Summary page opens:

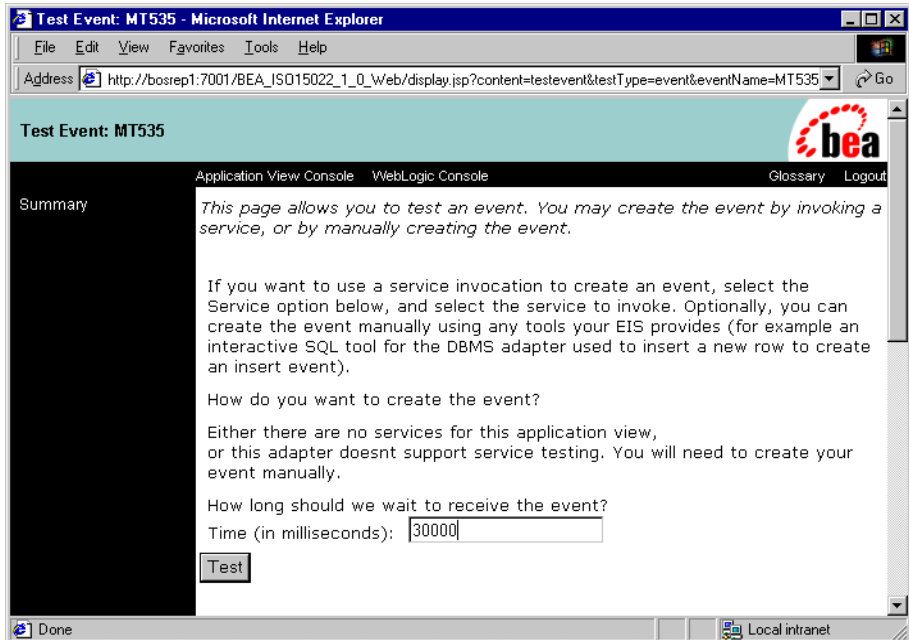
Figure 2-15 Application View Console: Summary Window



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

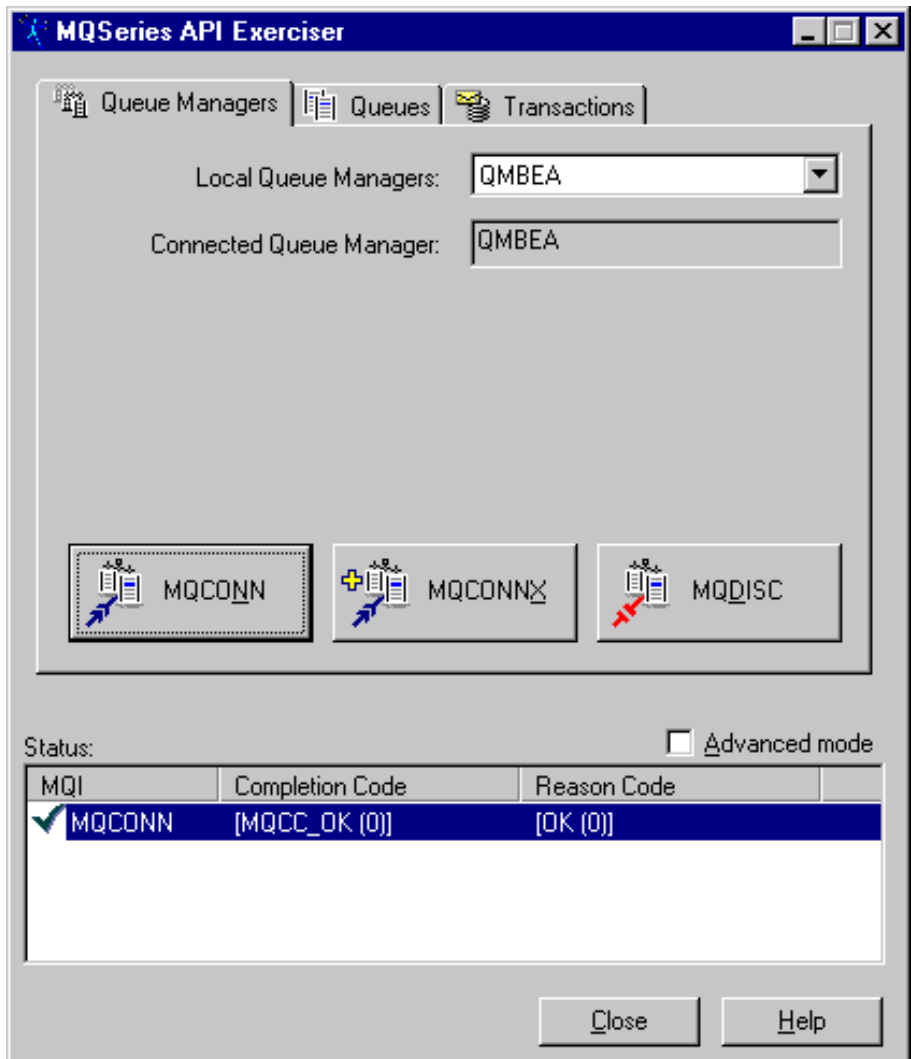
The Test Event page opens:

Figure 2-16 Application View Console: Test Event Window



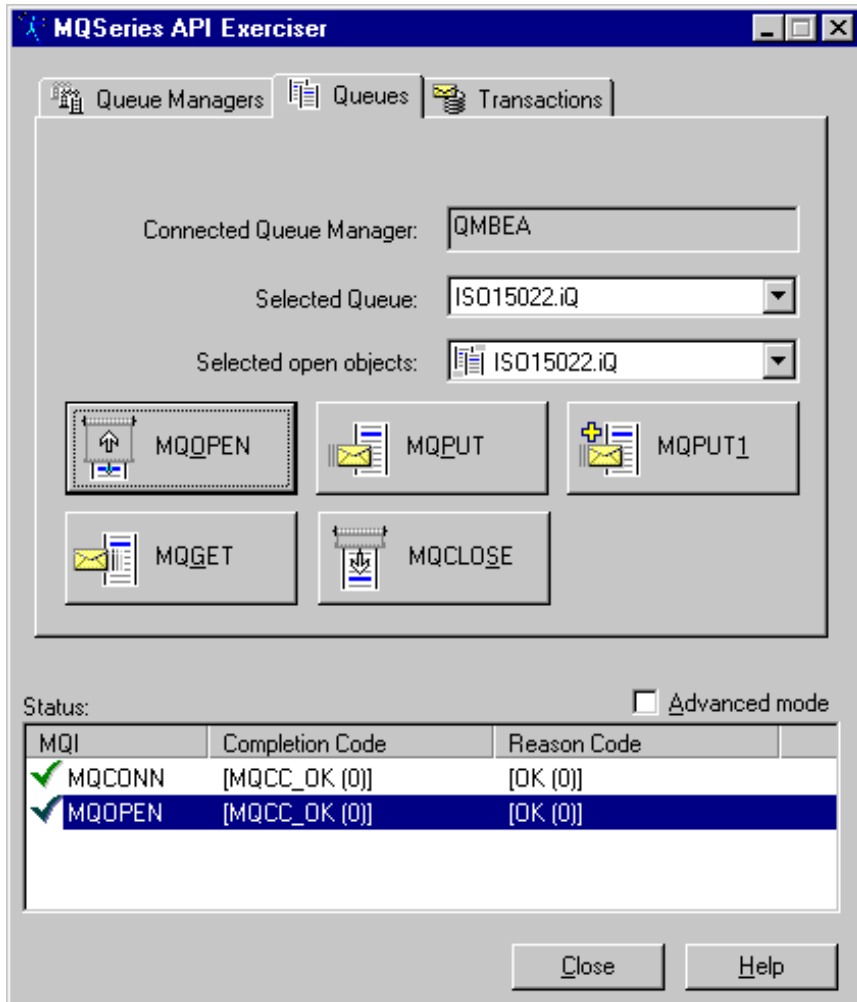
4. Enter 3000 (or a higher value) in the Time box. This provides a 30-second period during which, in the following step, you can access the IBM MQSeries API Exerciser (or your favorite utility) to manually invoke a request from MQSeries to your event adapter.
5. Open the MQSeries API Exerciser and select the Queue Manager that you had assigned to the event adapter in the Add Event page of the Application View Console.

Figure 2-17 MQSeries API Exerciser: Queue Managers Tab



6. Click MQCONN.
7. Click the Queues tab.

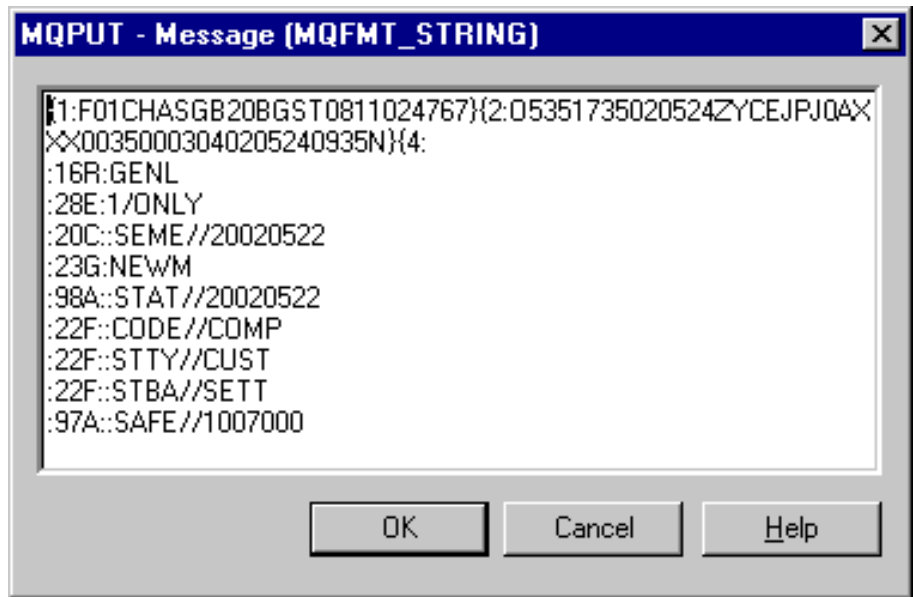
Figure 2-18 MQSeries API Exerciser: Queues Tab



8. On the Queues tab, in the Selected Queue drop-down list, select the queue that you had assigned to the event adapter in the in the Add Event page of the Application View Console.
9. Click MQOPEN.
10. Click MQPUT.

The MQPUT dialog box appears.

Figure 2-19 MQPUT



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

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

12. Examine the WebLogic Server Log for information about the event's activity.

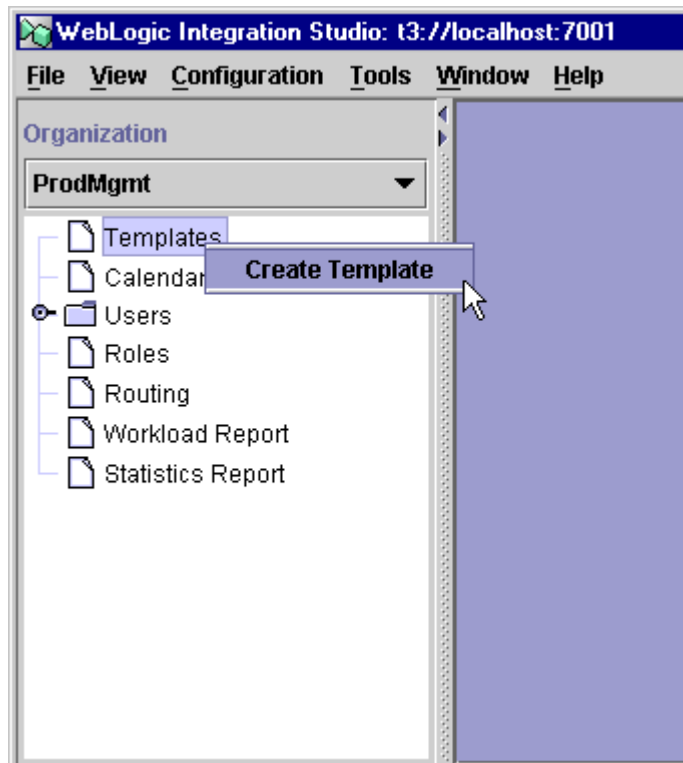
Otherwise, you have now confirmed that the event adapter application view is correctly configured and can receive events.

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:

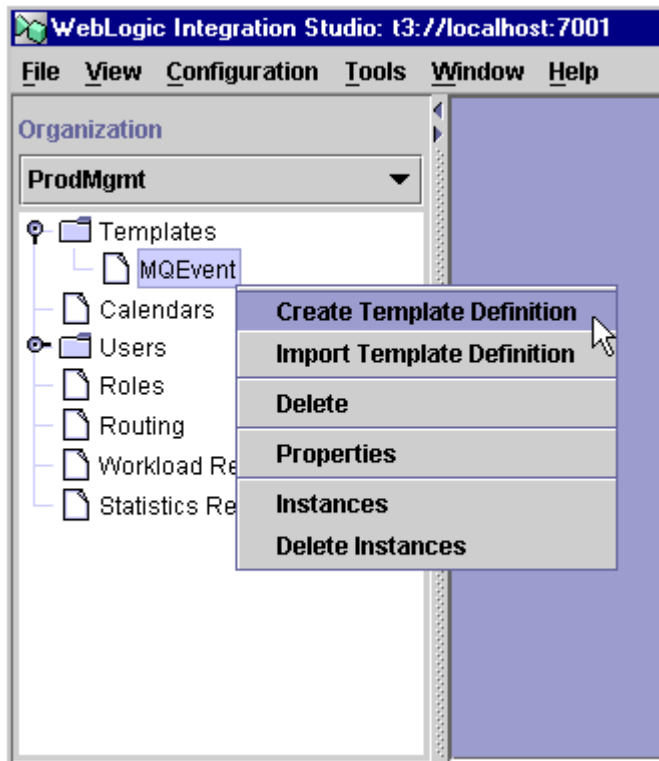
1. Start the WebLogic Integration Studio.
 - On a Windows system, choose Start→Programs→BEA WebLogic Platform 7.0→WebLogic Integration 7.0→Studio.
 - On a UNIX system, go to the WLI_HOME/bin directory and run the studio command.
2. Log on to the WebLogic Integration Studio.
3. In the Organization pane, choose an organization to create a new business process management workflow template.
4. Right-click Templates and select Create Template:

Figure 2-20 WebLogic Integration Studio - Create Template



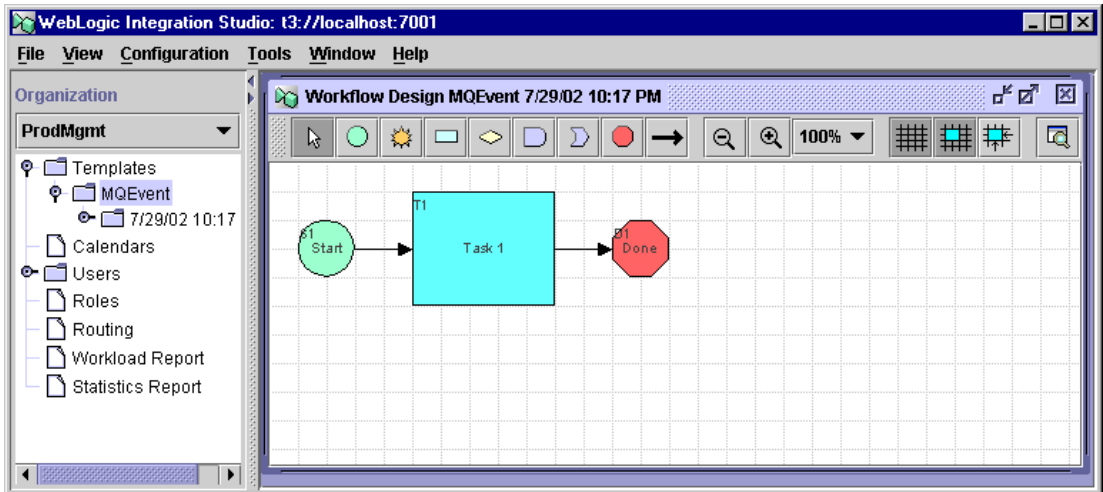
5. Right-click the new template and select Create Template Definition:

Figure 2-21 WebLogic Integration Studio - Create Template Definition



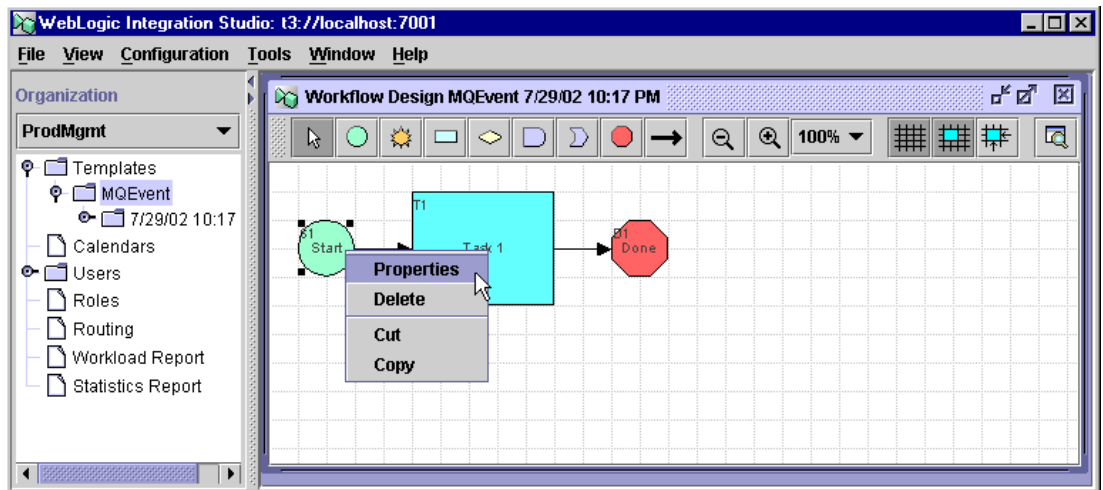
The template appears in WebLogic Integration Studio:

Figure 2-22 WebLogic Integration Studio - New Template



6. Right-click the Start node and select Properties:

Figure 2-23 WebLogic Integration Studio - Start Node Properties



The Start Properties dialog box appears.

Figure 2-24 WebLogic Integration Studio - Start Properties

Start Properties

Description
Start

☐ Timed ☐ Manual ☐ Called ☒ Event **AI Start** ▼

Event Explorer:
Root
 MQSeries
 MQTesting
 MQEvent
 OrderIn_csv
 MQService
 SAP

Name:
OrderIn

Description:

Condition:
 A+BQ

Event Document Variable:
 ▼

Refresh Tree View Definition

Start Organization
 ▼

☐ Use workflow expression

Variables Actions Next Notes

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

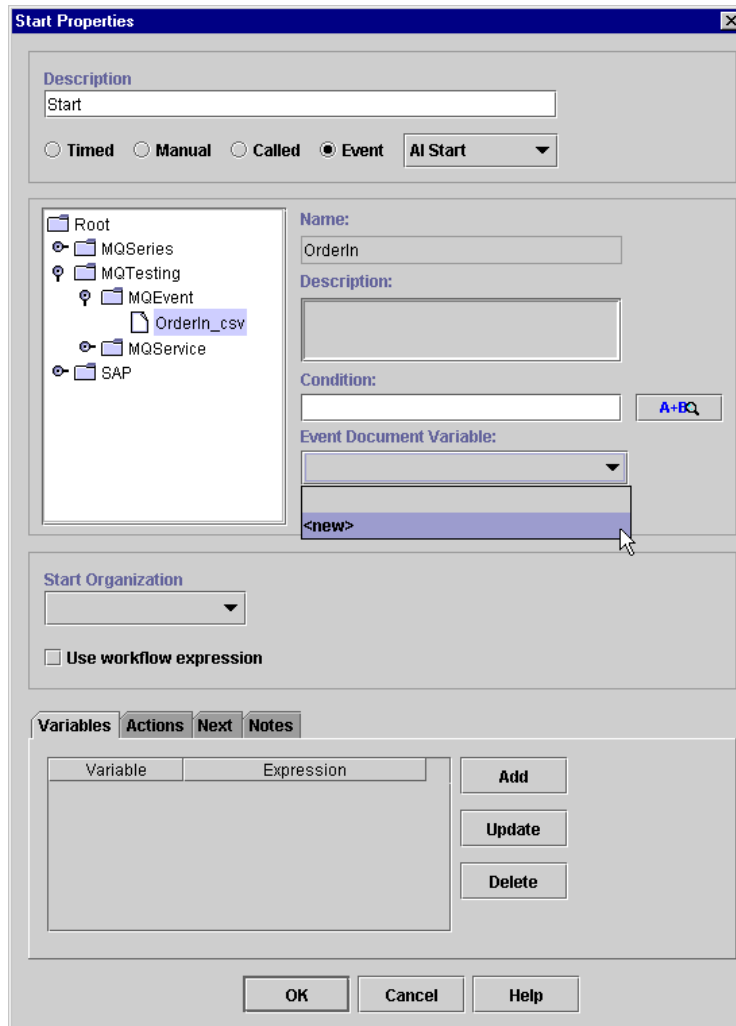
Add
Update
Delete

OK Cancel Help

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:

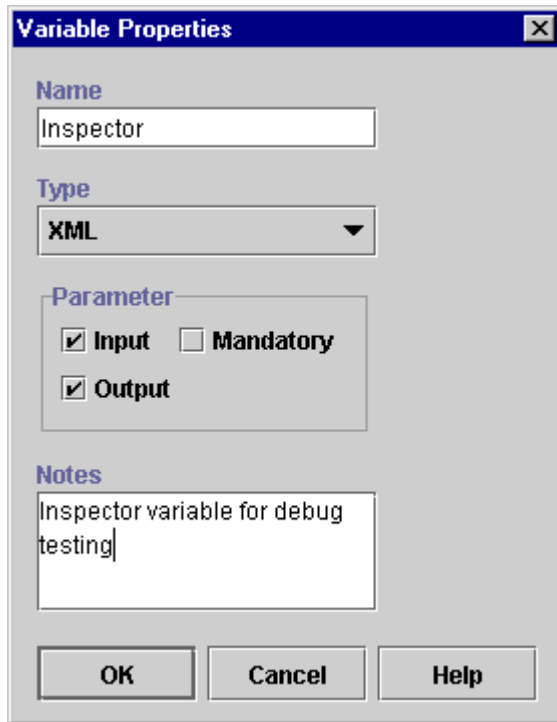
10. Select <new> from the Event Document Variable drop-down list:

Figure 2-25 Start Properties Dialog Box - New Document Variable



The Variable Properties dialog box appears.

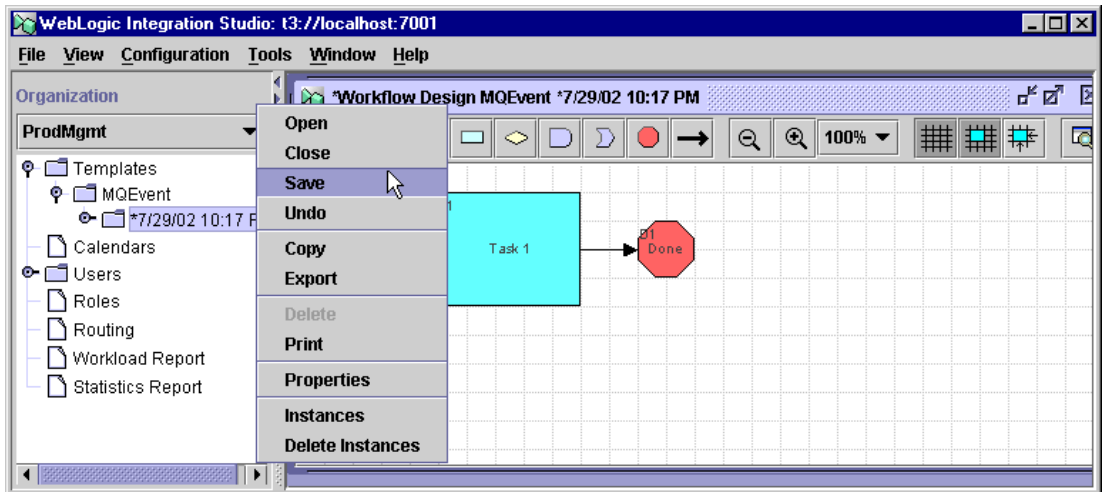
Figure 2-26 Variable Properties Dialog Box



The image shows a 'Variable Properties' dialog box with a blue title bar and a close button. It contains several sections: 'Name' with a text field containing 'Inspector'; 'Type' with a dropdown menu showing '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.

11. Type a name for the new variable.
12. Select the variable type XML.
13. Check the Input and Output options in the Parameter group:
14. Click OK.
15. Right-click the template in WebLogic Integration Studio's left pane and select Save:

Figure 2-27 WebLogic Integration Studio - Save Template



16. Right-click the event definition folder and select Properties.

The Template Definition dialog box appears.

Figure 2-28 Template Definition Dialog Box



The image shows a dialog box titled "Template Definition MQEvent". It has two tabs: "General" and "Exception Handlers". The "General" tab is selected. Inside the dialog, there is a "Workflow Label" text field with a search icon (A+BQ) to its right. Below this is a checkbox labeled "Active" which is checked. Underneath the "Active" checkbox are two date pickers: "Effective" (set to Jul 29, 2002) and "Expiry" (set to Dec 31, 2002). Below the date pickers is a checkbox labeled "Enable auditing" which is unchecked. At the bottom of the dialog is a "Notes" text area. At the very bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

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

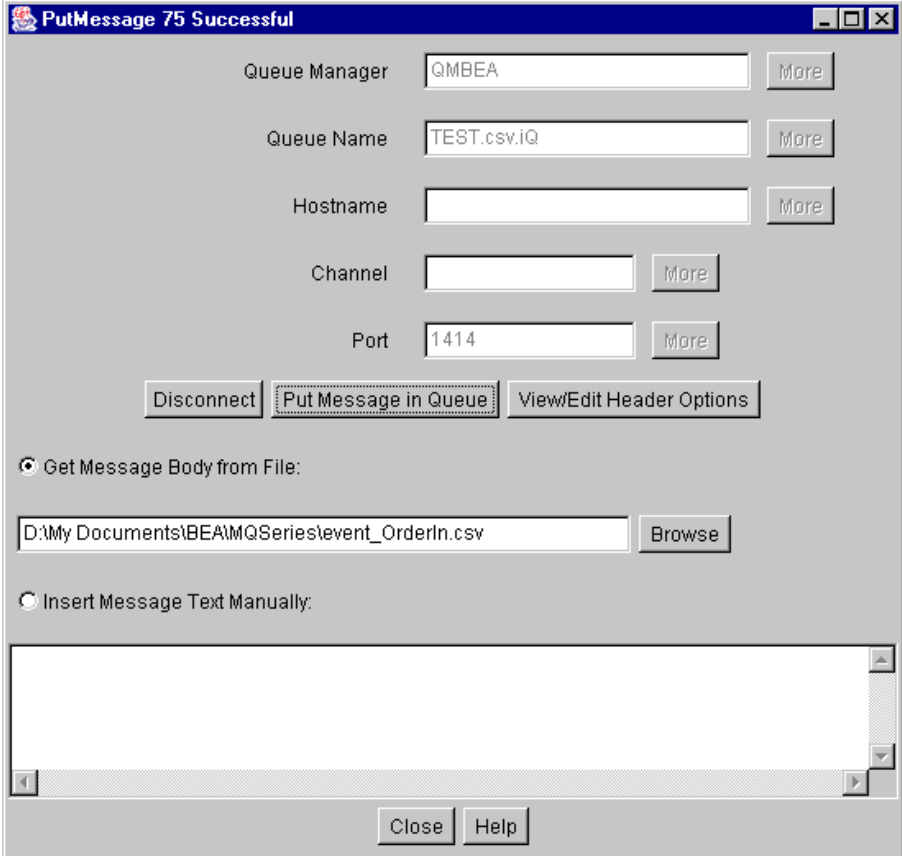
You may now initiate events from your Enterprise Information System.

For the BEA WebLogic Adapter for ISO15022, you can create events through a business application or through an MQSeries utility.

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

The PutMessage utility assists in writing messages into the queue.

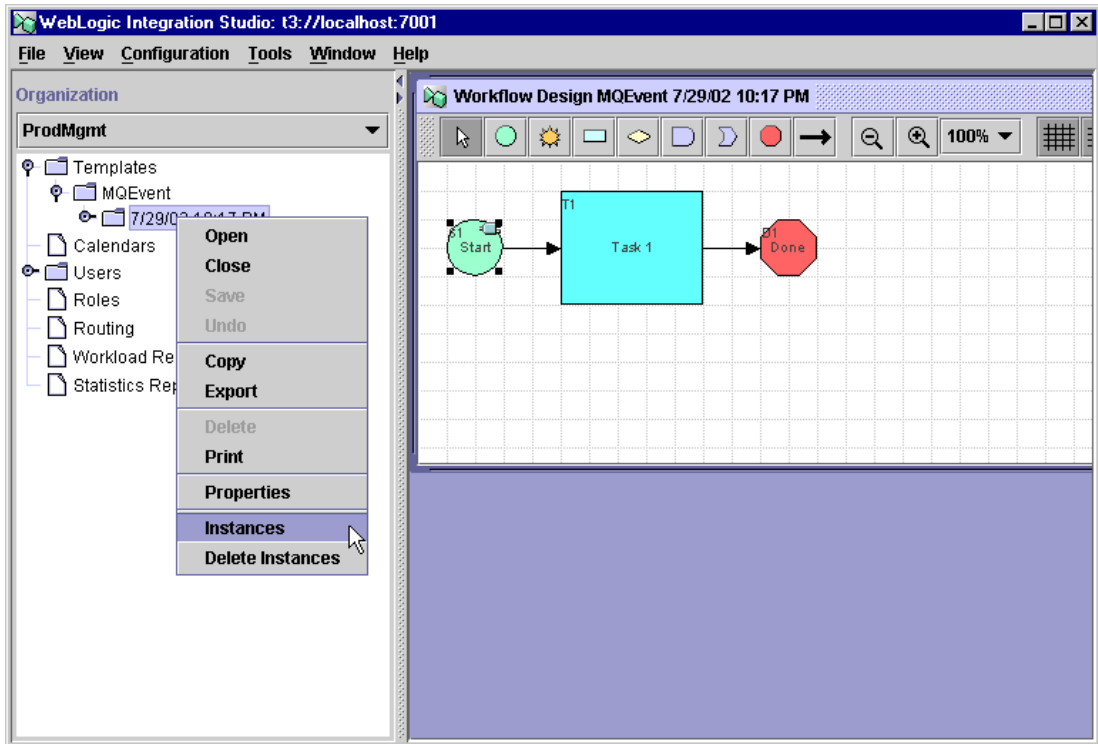
Figure 2-29 MQSeries PutMessage



The image shows a Windows-style dialog box titled "PutMessage 75 Successful". It contains several input fields and buttons. The "Queue Manager" field is set to "QMBEA", "Queue Name" is "TEST.csv.iQ", and "Port" is "1414". There are "More" buttons next to each of these fields. Below these fields are three buttons: "Disconnect", "Put Message in Queue" (which is highlighted with a dashed border), and "View/Edit Header Options". Underneath is a radio button labeled "Get Message Body from File:" which is selected. Below this is a text field containing the path "D:\My Documents\BEA\MQSeries\event_OrderIn.csv" and a "Browse" button. There is another radio button labeled "Insert Message Text Manually:" which is not selected. Below this is a large, empty text area with a scrollbar. At the bottom of the dialog are "Close" and "Help" buttons.

1. Enter free-form text or source the data from a file.
2. Then specify a queue manager, queue name, and port for your locally accessible MQ installation.
3. Select the message or message source file and click Put Message in Queue.
4. Return to WebLogic Integration Studio.

Figure 2-30 WebLogic Integration Studio - Event Definition Folder



5. Right-click the event definition folder and select Instances:

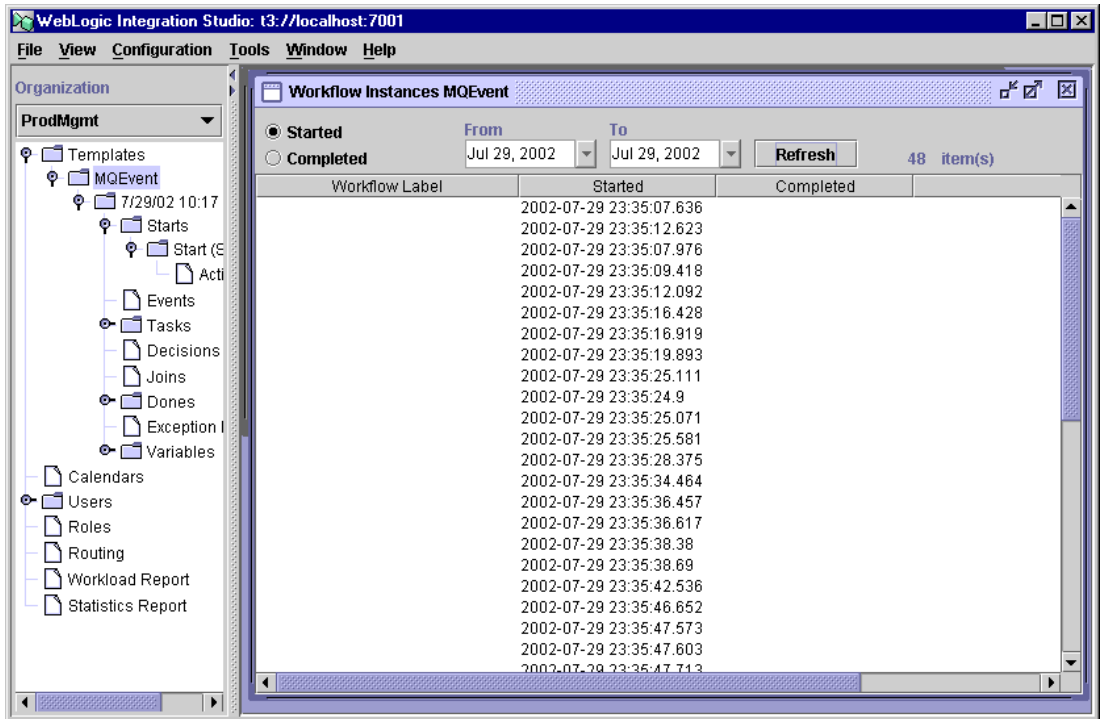
The Workflow Instances for your event definition appear.

You can now track execution of your workflow.

6. Select Started and click Refresh.

You should see a list of started work flows:

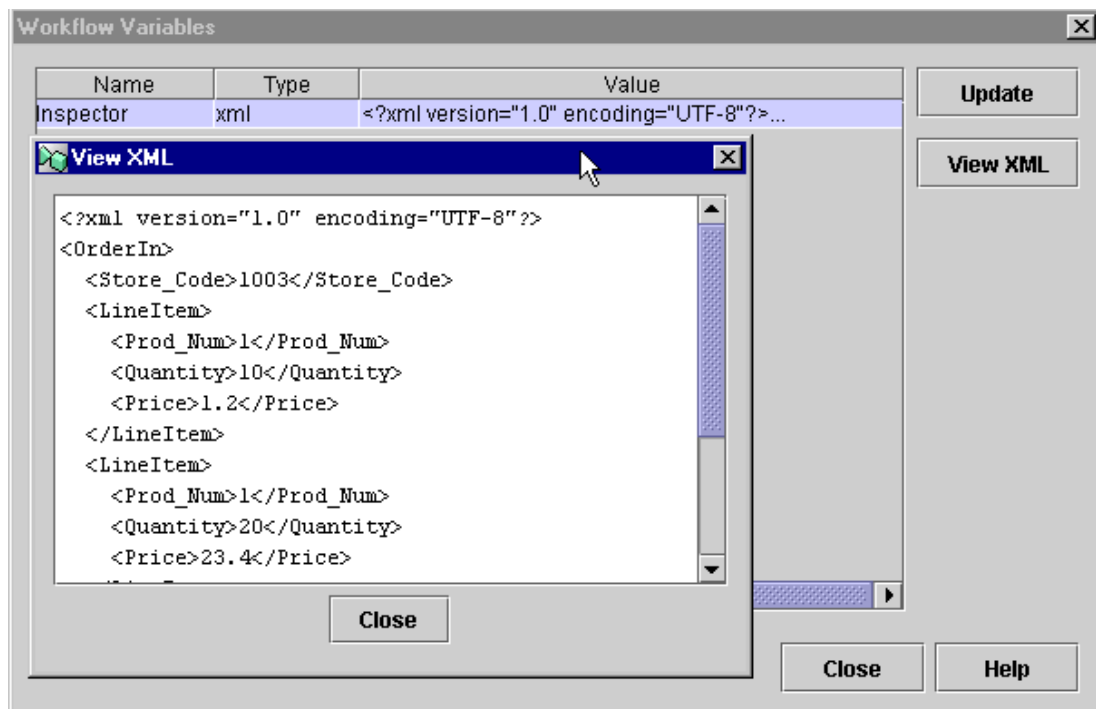
Figure 2-31 WebLogic Integration Studio - Workflow Instances



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

The Workflow Variables dialog box appears.

Figure 2-32 Workflow Variables Dialog Box



8. Click View XML to see the entire contents of the workflow document.

3 Creating and Configuring a Service Adapter

The service adapter for ISO15022 is WebLogic Integration's interface to SWIFT ISO 15022 message systems. It enables your business processes to write output in SWIFT ISO15022 or 7775 format. This section contains the following topics:

- [How to Create a Service Adapter Application View](#)
- [How to Configure a Service Adapter Application View](#)
- [Testing the BEA Service Adapter for ISO15022](#)

How to Create a Service Adapter Application View

The service adapter for ISO15022 is configured for all services that must send data in SWIFT format (either from or to 15022 format).

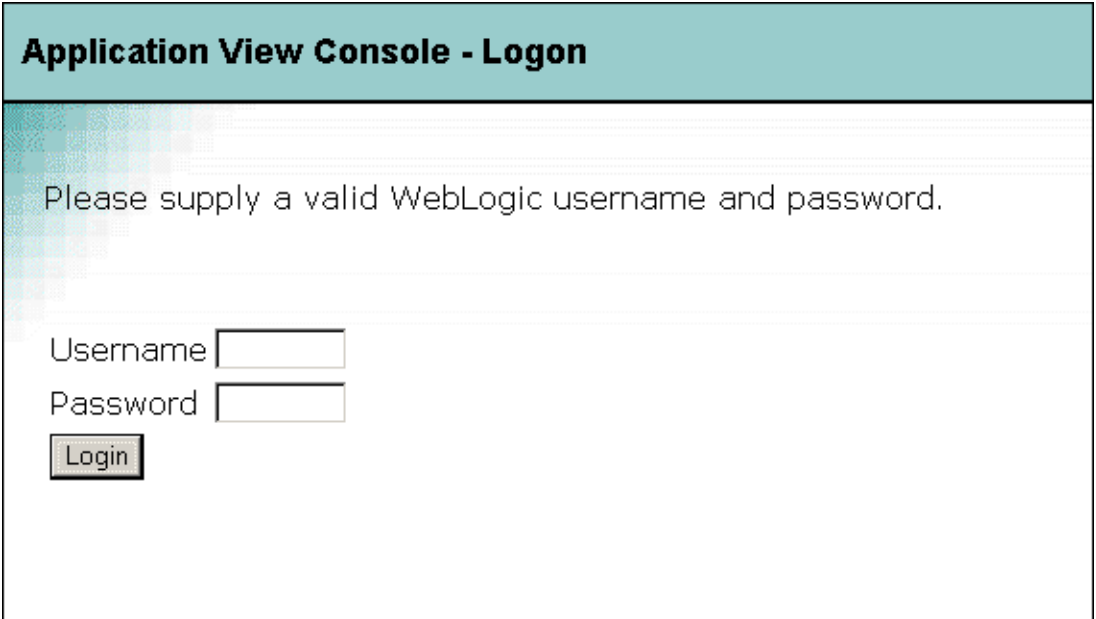
1. Open the Application View Console, which is found at the following location:

`http://host:port/wlai`

Here, *host* is the TCP/IP address or DNS name where WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The default port at the time of installation is 7001.

2. If prompted, enter a user name and password, as shown in the following figure.

Figure 3-1 Application View Logon



Application View Console - Logon

Please supply a valid WebLogic username and password.

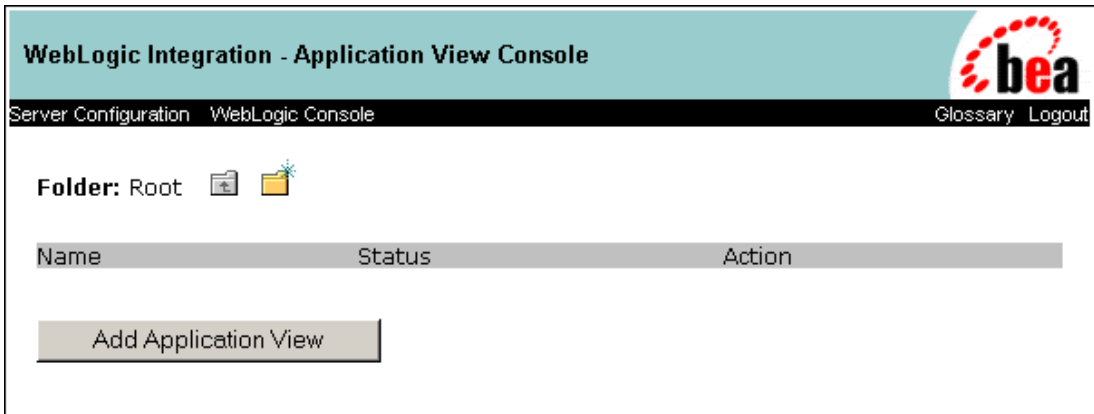
Username

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 ISO15022 Installation and Configuration Guide*.

3. Click Login.

Figure 3-2 Application View Console



4. Click Add Application View to create an application view for the adapter. The Define New Application View dialog box opens. An application view enables a set of business processes for this adapter's target EIS application. For more information, 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

3 *Creating and Configuring a Service Adapter*


The Define New Application View page opens.

Figure 3-3 Define New Application View

Application View Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

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

Define New Application View  Glossary Logout

This page allows you to define a new application view

Folder: **ISO15022**

Application View Name:

Description:

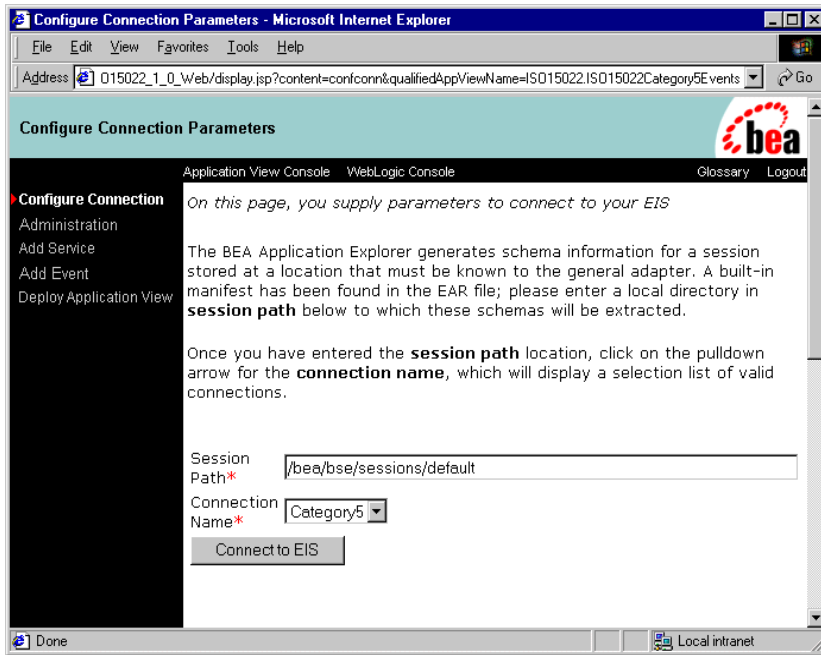
Associated Adapter:

Done Local intranet

5. Type a name and description for the application view.
6. Select BEA_ISO15022_1_0 from the Associated Adapter list.
7. Click OK.

The Configure Connection Parameters page opens.

Figure 3-4 Configure Connection Parameters



8. Type the name of the BEA WebLogic Adapter for ISO15022 session base directory in the Session path box.

This directory holds your SWIFT schema information and contains the subdirectory `ISO15022/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 `/bea/bse/sessions/default/ISO15022/QMBEA`.

9. From the Connection name drop-down list, select the session name (also known as the connection name).

10. Click Connect to EIS. The Application View Administration page opens.

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

How to Configure a Service Adapter Application View

After you create and configure an application view, you can add services that support the application's functions. For information on creating the application view, see [How to Create a Service Adapter Application View](#).

To add a service to an application view:

1. If it is not already open, open the application view to be modified.

For more information, see “Editing an Application View” in “Defining an Application View” in *Using Application Integration*:

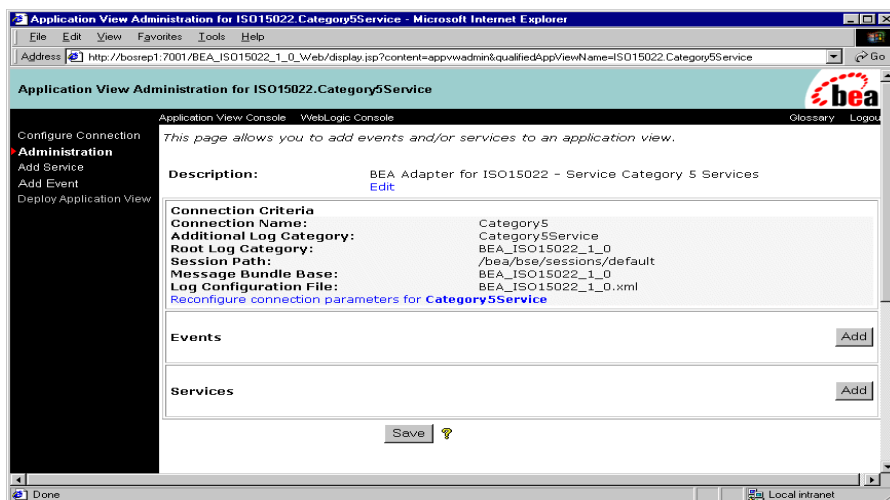
- For WebLogic Integration 7.0, see <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>
- For WebLogic Integration 2.1, see http://edocs.bea.com/wlintegration/v2_1sp/aiuser/2usrdef.htm

2. If the application view is deployed, you must undeploy it before adding the service. See “Optional Step: Undeploying an Application View” in “Defining an Application View” at the URL referenced in the previous step.

3. In the left pane, click Administration from the Configure Connection list.

The Application View Administration window opens.

Figure 3-5 Application View Administration for ISO15022.Category5Service



1. From the Services pane of the Application View Administration page, select Add.
The Add Service page opens.

Figure 3-6 Add Service Window

2. Enter the required properties of XMLtoXML as described in the table that follows.

Table 3-1 XML to XML Service Properties

Property	Description	Type	Sample Value	Element
Transform Type* (*Required)	The name of the transformation template file. Select from the drop-down of available types. For more information, see Chapter 4, “Transforming Document Formats.”	drop down		<in_xmlg>

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

For MQSeries, the following figure shows the parameters:

Figure 3-7 Add Service Window - MQEmit

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: * MT541

Select: MQEmit_ISO15022

Queue Manager*	QMBEA
Queue Name*	ISO15022.IQ
Correlation Id	
Retry Duration	
Retry Interval	
Recycle Interval	
MQ Client Host	
MQ Client Port	
MQ Client Channel	
Execution Time Limit	
Polling Interval	

schema: 541

settings

Trace on/off	<input type="checkbox"/>
Logging on/off	<input type="checkbox"/>
Deep Debug on/off	<input type="checkbox"/>
Maximum Log Size (kb)	500
root to transform template directory	L:/SCHEMAS/ISO15022/category6/te
root to XML Style sheet directory	L:/SCHEMAS/ISO15022/category6/te

Add

3 *Creating and Configuring a Service Adapter*

3. Enter the required properties of MQSeries as described in the following MQSeries Properties Table.

The properties correspond to the MQSeries communication settings that the service adapter uses to communicate with MQSeries when writing messages to the queues. They also correspond to the transform options described in [Chapter 4, “Transforming Document Formats.”](#)

Table 3-2 MQSeries Service Properties

Property	Description	Type	Sample Value	Element
Queue Manager* (*Required)	Name of the MQSeries Queue Manager to be used.	string	QMBEA	<manager>
Queue Name* (*Required)	Queue on which request documents are received.	string		<queue>
Correlation Id	The correlation ID to set in the MQSeries message header.	duration		<correlid>
Retry Duration	Maximum time that a document can remain in the retry pending queue.	duration		<duration>
Retry Interval	Interval between retrying pending requests.	duration		<retry>
Recycle Interval	Interval between retrying successful requests.	duration		<recycle>
MQ Client Host	For MQ Client only. Host on which MQ Server is located.	string		<host>
MQ Client Port	For MQ Client only. Port number to connect to an MQ Server.	integer		<port>

Table 3-2 MQSeries Service Properties (Continued)

Property	Description	Type	Sample Value	Element
MQ Client Channel	For MQ Client only. Channel between an MQ Client and MQ Server.	string		<channel>
Execution Time Limit	The length of time in seconds before execution is terminated.	duration		<maxlife>
Polling Interval	The time in milliseconds.	duration		<timeout>

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

For File, the following figure shows the applicable parameters:

Figure 3-8 Add Service - File

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:* MT541

Select: FileEmitter_ISO15022

directory*

output file name/mask*

schema: 500_to_502

settings

Trace on/off	<input type="checkbox"/>
Logging on/off	<input type="checkbox"/>
Deep Debug on/off	<input type="checkbox"/>
Maximum Log Size (kb)	500
root to transform template directory	L:/SCHEMAS/ISO15022/category6/te
root to XML Style sheet directory	L:/SCHEMAS/ISO15022/category6/te

Table 3-3 Service Properties - File

Setting	Meaning/Properties
directory* (*Required)	Type/Value: Directory Path Description: Directory to which output messages are emitted.
output file name/mask* (*Required)	Type/Value: String Description: The output file name (can contain a '*'), which gets expanded to a timestamp. A pound symbol can be used as a mask for a sequence count. Each pound symbol represents a whole number integer value. For example, File## counts up to 99 before restarting at 0, File### counts up to 999 before restarting at 0, and so on.

For FTP, the following figure shows the applicable parameters:

Figure 3-9 Add Service - FTP

Add Service

Application View Console WebLogic Console

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

Unique Service Name:* MT541

Select: FTPEmitter_ISO15022

Host name*	
Port number	
User Id*	
Password*	
destination*	
output file name/mask*	
Retry Interval	
Maxtries	

schema: 541

settings

Trace on/off	<input type="checkbox"/>
Logging on/off	<input type="checkbox"/>
Deep Debug on/off	<input type="checkbox"/>
Maximum Log Size (kb)	500
root to transform template directory	L:/SCHEMAS/ISO15022/category6/te
root to XML Style sheet directory	L:/SCHEMAS/ISO15022/category6/te

Add

Table 3-4 Service Properties - FTP

Setting	Meaning/Properties
Host name* (*Required)	Type/Value: String Description: FTP target system.
Port number	Type/Value: Numeric Description: FTP target system port (leave empty for FTP default).

3 *Creating and Configuring a Service Adapter*

Table 3-4 Service Properties - FTP (Continued)

Setting	Meaning/Properties
User ID* (*Required)	Type/Value: String Description: User account ID to use when connecting to the protocol host.
Password* (*Required)	Type/Value: String Description: Password for the user account to use when connecting to the protocol host.
destination* (*Required)	Type/Value: String Description: Directory to address on the FTP target system.
output file name/mask	Type/Value: String Description: The output file name (can contain a '*'), which gets expanded to a timestamp.
Retry Interval	Type/Value: Retry interval duration in xxH:xxM:xxS format. (for example, 1H:2M:3S, which is 1 hour 2 minutes and 3 seconds) Description: The maximum wait interval between retries when a connection fails.
Maxtries	Type/Value: String Description: Number of retries for a failed attempt to write.

4. Click Add and continue to the Deploy Application View page.

Figure 3-10 Deploy Application View Window

Deploy Application View ISO15022.ISO15022Category5Events to Server

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

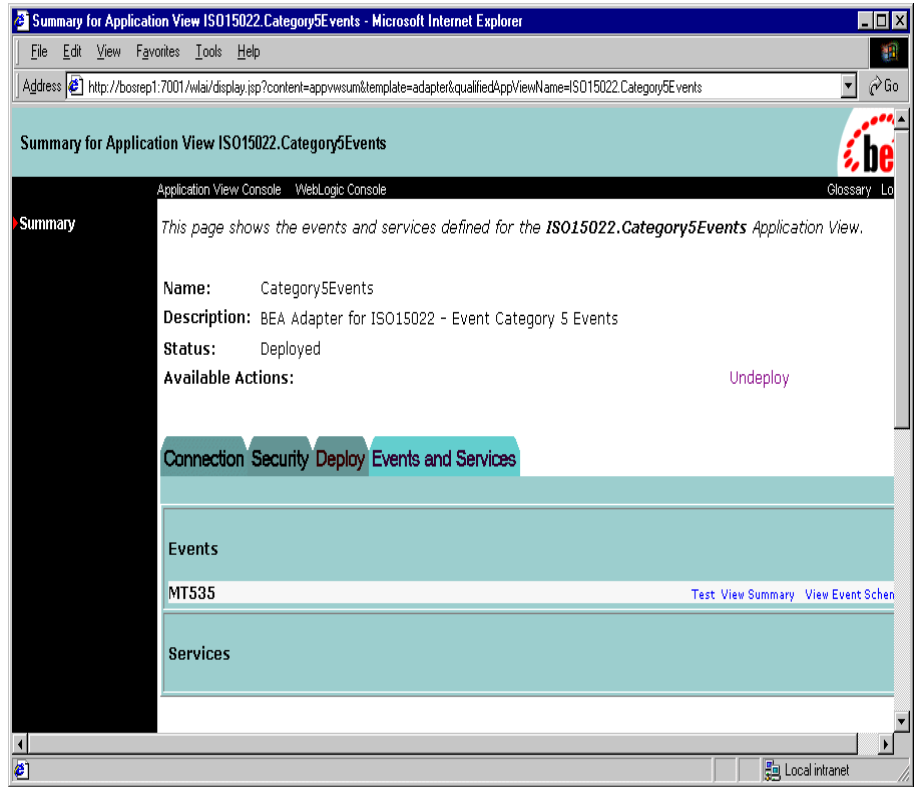
[Restrict Access to ISO15022Category5Events using J2EE Security](#)

☒ Deploy persistently?

5. Make any changes that you require to the Deploy Application View page and click Deploy.

The Summary for Application View page opens on successful completion of the Application View Deploy.

Figure 3-11 Summary for Application View



Testing the BEA Service Adapter for ISO15022

The service adapter emits a document to MQSeries and returns an emit status. You can validate the arrival of the document on the queue by browsing the MQSeries resources through IBM-supplied or custom tools. For example, on a Windows platform, you could use MQSeries Explorer, a Microsoft MMC plug-in. Using MQSeries Explorer 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.

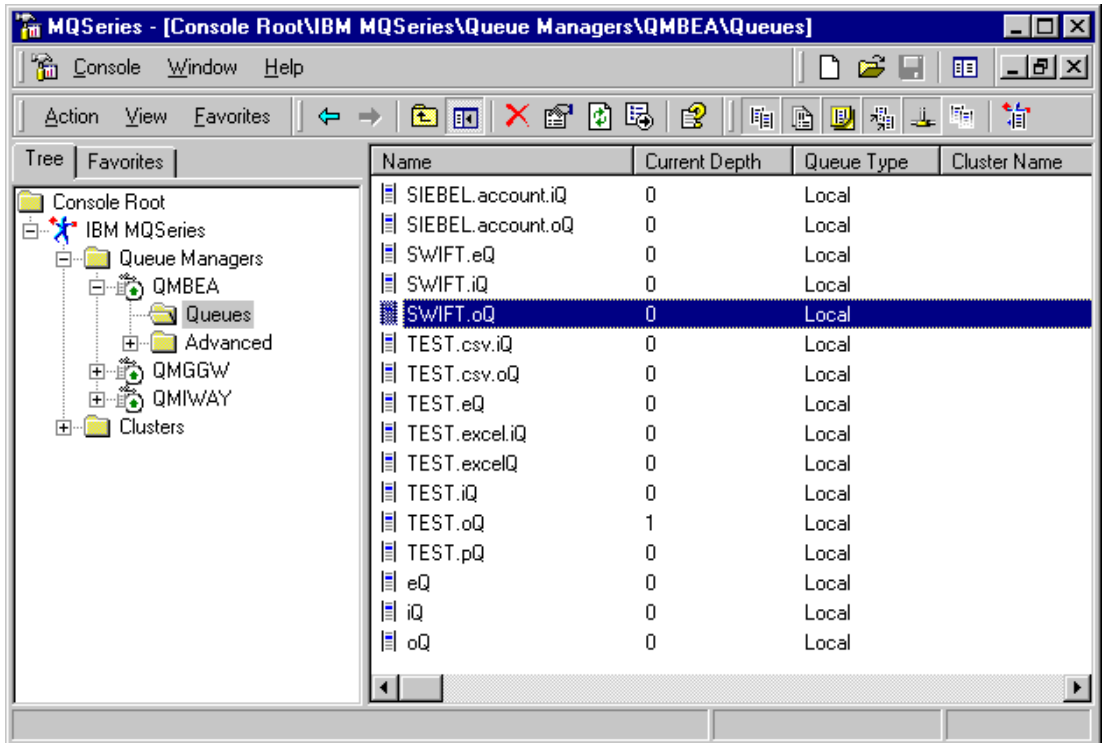
1. Confirm that the queue to which the service adapter sends messages is empty.

For example, using MQSeries Explorer, browse the applicable queue and check the current queue depth, ensuring that it is zero.

- a. To delete the existing queue messages, right-click and select All Tasks
- b. Select Clear Messages.

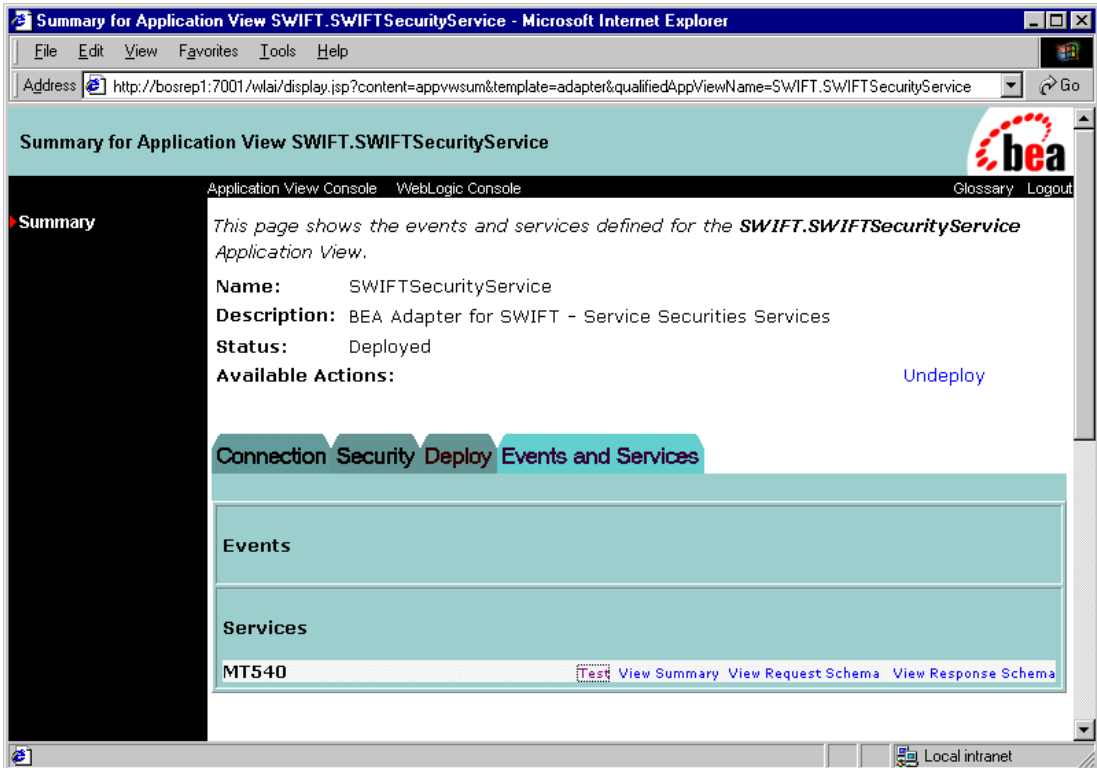
3 *Creating and Configuring a Service Adapter*

Figure 3-12 Confirming an Empty Output Queue



2. From the Summary for Application View page, click Test.

Figure 3-13 Summary for Application View Window



3. Enter a sample XML document that matches the request schema for the configured service.

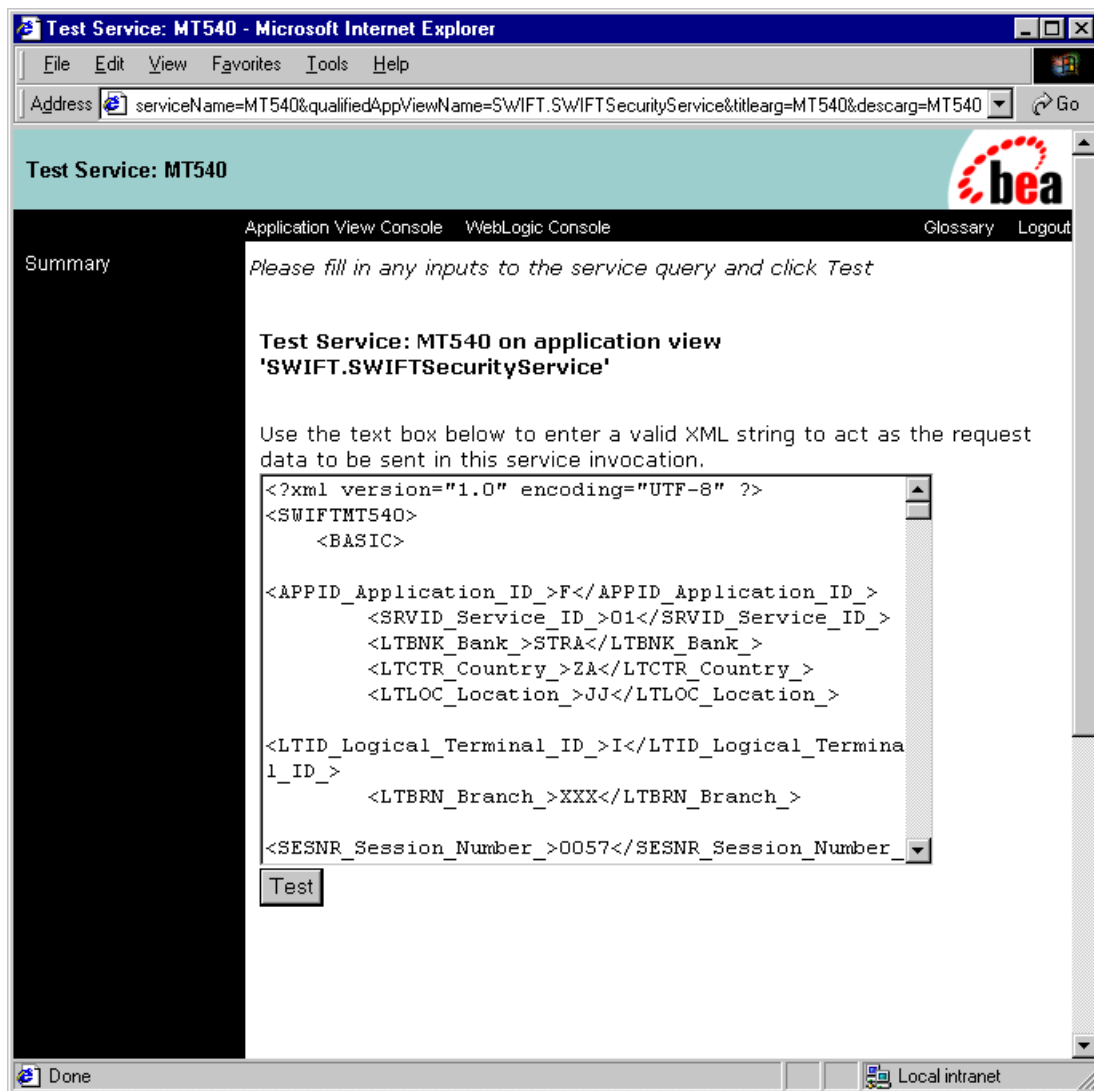
For example, the SWIFT MT540 request schema has an instance document similar to the following.

Figure 3-14 Instance Document

```
<?xml version="1.0" encoding="UTF-8" ?>
<SWIFTMT540>
  <BASIC>
    <APPID_Application_ID_>F</APPID_Application_ID_>
    <SRVID_Service_ID_>01</SRVID_Service_ID_>
    <LTBNK_Bank_>STRA</LTBNK_Bank_>
    ...
    lines of document omitted
    ...
  </BASIC>
</SWIFTMT540>
```

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

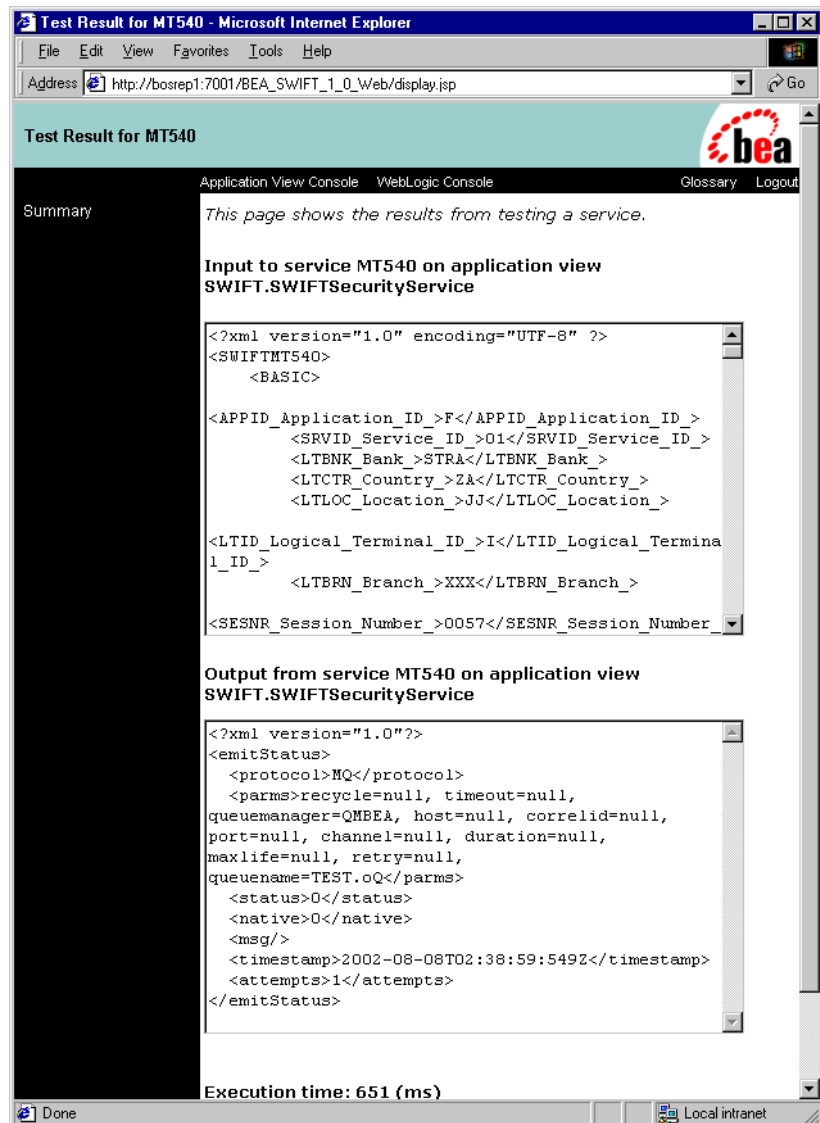
Figure 3-15 Test Service Window



5. Select Test to send the request through the ISO15022 service adapter to the IBM MQSeries Queue.

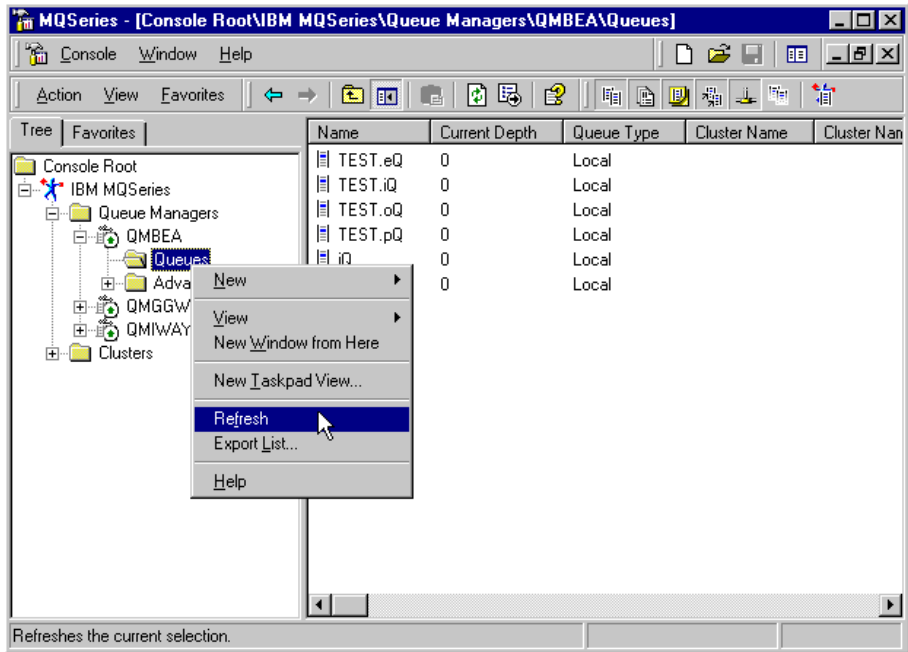
The response document should indicate the success of emitting to MQSeries. See the following sample response.

Figure 3-16 Test Result Window - Request Successfully Sent



After transformation and formatting, the document should arrive on the MQSeries queue.

Figure 3-17 MQSeries Queue Window

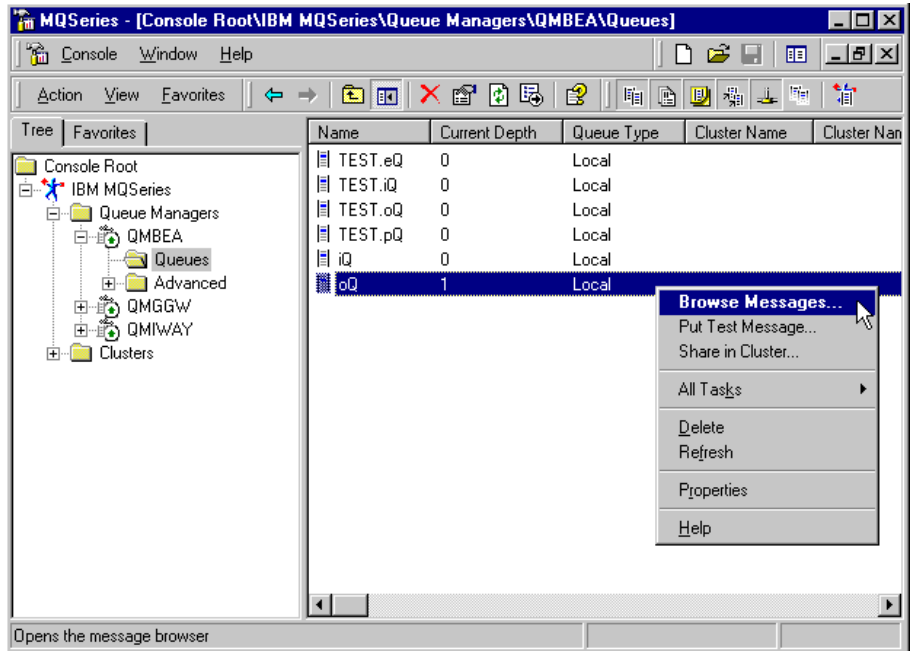


6. Check for the output document on the queue designated in configuring the service.
 - a. Using MQSeries Explorer, browse to the appropriate queue.
 - b. Select Refresh.

The Current Depth should increase incrementally.

Current Depth should be at one if you cleared all messages first.

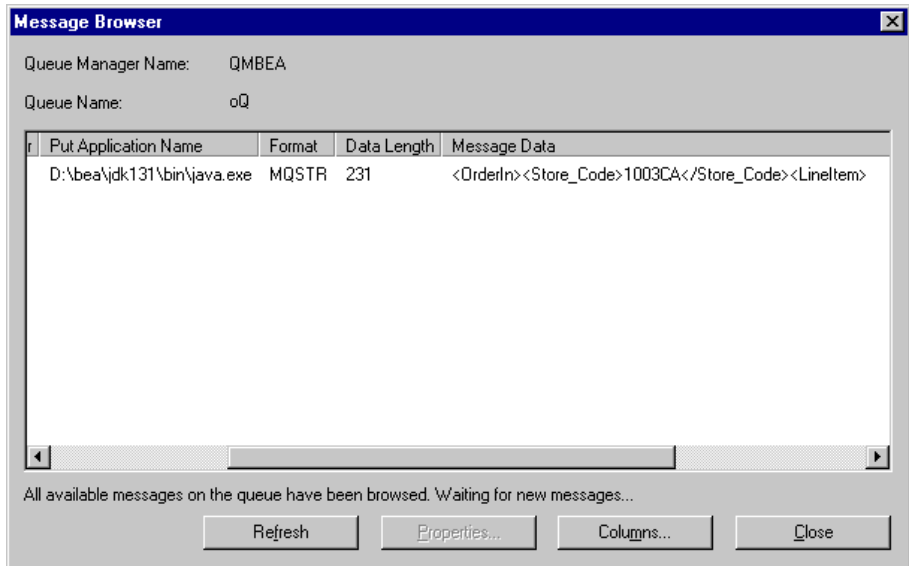
Figure 3-18 MQSeries Queue Window - Browse Messages



7. Select the appropriate queue.
8. Right-click and select Browse Messages.

You should see a window similar to the following Message Browser with a column containing the data that the service adapter sent to MQSeries:

Figure 3-19 Message Browser Window



4 Transforming Document Formats

This section describes how events are incorporated into workflow design. It includes the following topic:

- [Kinds of Transformation](#)
- [SWIFT to XML Transformations](#)
- [XML to XML Transformations](#)
- [XML to SWIFT Transformations](#)

Documents within WebLogic Integration are encoded in XML. However, you may need to receive and generate non-XML data. BEA adapters support inbound transformations for the event adapter and outbound transformations for the service adapter. This section describes the transformation options available to you.

The BEA WebLogic Adapter for SWIFT provides SWIFT to XML transformation in the event adapter and XML to SWIFT and XML to XML transforms in the service adapter.

Kinds of Transformation

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 is one transformation phase in the process of an event adapter sending a message from an Enterprise Information System to business process management workflows.

- SWIFT format to XML.

There are two transformation phases in the process of a service adapter sending a message from business process management workflows to a SWIFT message handling system.

- XML to XML.
- XML to SWIFT format.

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

SWIFT to XML Transformations

An event adapter that interfaces with SWIFT data sources must be capable of converting that data to XML for processing by business process management workflows. This conversion entails “pre-parsing” the data into XML. The XML is then parsed for processing.

SWIFT format messages arriving at the event adapter are processed by the SWIFT pre-parser, which determines the SWIFT message type and applies the correct SWIFT to XML format conversion.

XML to XML Transformations

A service adapter may be required to convert data from one type of XML document to another. You can choose the types of transform to accomplish this conversion by choosing the appropriate schema.

For example, in the following XMLtoXML service, the type of transform required is specified as 521_to_541.

Figure 4-1 Application View Console - Add Service

Edit Service

Application View Console WebLogic Console

On this page, you edit service properties.

Unique Service Name: XMLtoXML

Select: XMLtoXML

Transform_Type: 521_to_541

schema: 521_to_541

settings

Trace on/off	<input checked="" type="checkbox"/>
Logging on/off	<input type="checkbox"/>
Deep Debug on/off	<input type="checkbox"/>
Maximum Log Size (kb)	500
root to transform template directory	C:/Program Files/BEA System
root to XML Style sheet directory	C:/Program Files/BEA System

Apply

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

Table 4-1 XML to XML Transform Parameters

Parameter	Value/Description/Example
Transform Type	Type/Value: drop-down list Description: The name of the transformation template file. Select from the drop-down of available types.

XML to SWIFT Transformations

A service adapter that interfaces with SWIFT data sources must be able to convert XML documents (for example, from business process management) to SWIFT standard format. This conversion entails the pre-emit conversion of the data into the SWIFT, non-XML format, which is then emitted to the SWIFT message handling system (MHS).

5 Acknowledgement Handling

This section describes the process of acknowledging a document after it has passed through validation. It includes the following topics:

- [Acknowledgement Processing](#)
- [Documents, Validation, and Acknowledgement](#)
- [Acknowledgement Agent](#)
- [Acknowledgement Message Handling](#)

Documents received by the BEA WebLogic Adapter for SWIFT are processed in stages that include preparse, validate, transform, agent execute, and pre-emit. At any phase, the document may generate errors and may or may not pass specific validation rules. The validation engine and document validation rules are described in full in [Appendix A, “BEA WebLogic Adapter for ISO15022 Rules System.”](#)

Acknowledgement Processing

The acknowledgement process indicates the receipt and validity of a received document. The features which support the acknowledgement process are:

- validation
- document tree
- acknowledgement agent

Validation

Validation is a specific stage in processing the document that occurs immediately after the document arrives and is available in XML format (that is, after the XML structure is available but before any other processing). The process of validation and the rules used in validating a document are described in [Appendix A, “BEA WebLogic Adapter for ISO15022 Rules System.”](#)

Document Tree

The document tree is the adapter's representation of the XML document. The tree is used during document processing and stores additional document or element level information. Validation errors are stored in the document tree and are available to the acknowledgement agent.

Acknowledgement Agent

The acknowledgement agent is responsible for determining what processing should occur for a document, as represented in memory by the document tree, and contains zero, one, or more validation errors. The agent determines what constitutes a good document to which an "ACK," or acknowledgement, should be sent. It may also determine what constitutes a bad document to which a "NAK," or non-acknowledgement, should be sent. The agent also determines the content of the ACK and the NAK.

Documents, Validation, and Acknowledgement

Documents proceed from the adapter event listener to the emitter or event poster, and are processed in stages.

With respect to validation and acknowledgement, the above document life cycle has the following characteristics:

- Validation occurs as soon as the document has been converted into XML.
- Validation comprises both structural validation (described in dictionaries) and content & network validation (described in `Rules.xml` files).
- The validation processor (class) is defined at the document level or at the rule level. (See [Appendix A, "BEA WebLogic Adapter for ISO15022 Rules System"](#))

for a description.) An example of a validation processor is `XDSWIFTRules.class`.

- Validation (particular `Dictionary` and `Rules.xml`) is tied to a specific XML document type (as defined in `Deploy.xml`).
- If validation is defined for an XML document, an acknowledgement agent is added to the agent vector (first in line) for processing during the document's agent execution phase. See [“Acknowledgement Agent” on page 5-4](#) for a discussion of acknowledgement agent determination.
- Validation processing adds error elements to the document tree.
- The acknowledgement agent processes the document using its document tree including any validation errors added during the previous validation phase.
- The output of the acknowledgement agent is independent of the output of the document agent (they have different schema, separate threads of execution, and so on).
- Validation errors and document output are independent of one another. In other words, a document may fail validation rules and have acknowledgements (ACK or NAK) generated in the acknowledgement agent. However, the document is still passed to its agent and sent to its output unless specific actions are taken (that is, logic is coded). For example, if the copy agent is defined for the particular document life cycle, the document is passed out and posted to the event router even if there are errors.
- Custom agents may be coded to alter behavior when validation errors are present in the document tree.

Acknowledgement Agent


The acknowledgement agent is defined by an `<ackagent>` entry. It can be defined at the event or service level, in the document section of the `Deploy.xml`, or as an attribute in the root element of the `Rules.xml` file. The search order is as follows:

1. The `<document>` section of the `deploy.xml`.
2. An attribute of the root element of the `Rules.xml` file.

3. The event or service listener definition using the Application View Console.

For the BEA WebLogic Adapter for SWIFT, the default ackagent is the XDSWIFTACKAgent. This is set or changed on the event or service configuration screen.

Figure 5-1 Edit Event Window

Edit Event


Application View Console WebLogic Console

[Configure Connection](#)
[Administration](#)
[Add Service](#)
[Add Event](#)
[Deploy Application View](#)

Glossary Logout

On this page, you edit event properties.

Unique Event Name: *

Select:

XSLT Transform	<input type="text"/>
Queue Manager*	<input type="text"/>
Queue Name*	<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"/>
ackagent	XDSWIFTACKAgent

schema:

settings

Trace on/off	<input checked="" type="checkbox"/>
Logging on/off	<input checked="" type="checkbox"/>
Deep Debug on/off	<input checked="" type="checkbox"/>
Maximum Log Size (kb)	<input type="text" value="500"/>
root to transform template directory	L:/SCHEMAS/ISO15022/category6/te
root to XML Style sheet directory	L:/SCHEMAS/ISO15022/category6/te

Acknowledgement Message Handling

The validation engine performs the structural, content, and network validation rules defined in the document specific dictionaries and `rules.xml` files. The acknowledgement agent generates an acknowledgement message based on the document tree which is a composite of the original XML document tree and validation errors added by the validation engine. The results of the acknowledgement agent are dependent on the logic coded in the implementation class. For the BEA WebLogic Adapter for SWIFT, the default implementation class is the `XDSWIFTACKAgent.class`.

The following is a sample output with errors.

Listing 5-1 Sample Output

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

<eda>

    <error code="-103" source="validator"
timestamp="2002-08-08T17:37:34Z">

        Document failed validation: XD[FAIL] validation
error: checkList
[SWIFTMT541._541.E.E3.L_19A._19A.CURCD_Currency_Code_]: code is
missing

    </error>

</eda>
```

The following schema for the results of this acknowledgement is the `XDSWIFTACKAgent.xsd`.

Listing 5-2 XDSWIFTACKAgent.xsd

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <xs:element name="Error" type="xs:string"/>

  <xs:element name="SWIFTack">

    <xs:complexType>

      <xs:sequence>

        <xs:element ref="Error" minOccurs="0"
maxOccurs="unbounded" />

      </xs:sequence>

    </xs:complexType>

  </xs:element>

</xs:schema>
```

The acknowledgement message is generated separately from the document message. After the acknowledgement agent completes execution, two messages traversing the system attempt to be posted to the event router. For the message to be posted, an event must be registered in the application view with the acknowledgement schema.

Creating an Acknowledgement Event

In addition to the application view event created for the document, there must be an event created for the acknowledgement message generated by the acknowledgement agent. The following procedure creates an event for the acknowledgements generated by the XDSWIFTACKAgent.

1. Add an event in the WebLogic application view of the event adapter.

Figure 5-2 Add Event Window - Adding an ACK_NAK Event

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: *

Select:

XSLT Transform	<input type="text"/>
Queue Manager*	<input type="text" value="QMBEA"/>
Queue Name*	<input type="text" value="SWIFT.iQ"/>
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"/>
ackagent	<input type="text" value="XDSWIFTACKAgent"/>

schema:

settings

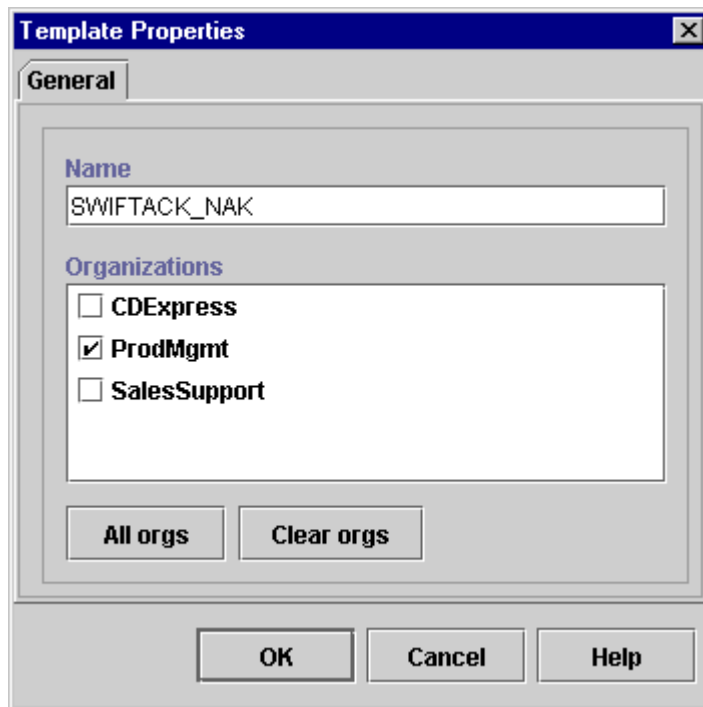
Trace on/off	<input type="checkbox"/>
Logging on/off	<input type="checkbox"/>
Deep Debug on/off	<input type="checkbox"/>
Maximum Log Size (kb)	<input type="text" value="500"/>
root to transform template directory	<input type="text" value="L:/SCHEMAS/ISO15022/category6/te"/>
root to XML Style sheet directory	<input type="text" value="L:/SCHEMAS/ISO15022/category6/te"/>

Note: The ackagent value is set to the desired acknowledgement agent (in this example, the XDSWIFTACKAgent). Additionally, there is an ACKNAK schema in the Schema drop down list box which handles both clean ACKs and error filled NAKs.

It is important that the event adapter's protocol settings (in this case, MQSeries based) are identical to those provided for the original event. If the settings are different, a separate event listener is created, and the two events (document and associated ACK or NAK) are not tied together. The ACKs or NAKs created by the document are not seen by the event or schema combination created in this section.

2. Add, continue, and deploy the application view.
3. From WebLogic Integration Studio, create a new workflow template in the Template Properties dialog box.

Figure 5-3 Template Properties Dialog Box - Creating a Workflow Template



4. Give the template a name to indicate this workflow is for the acknowledgement message.
5. Click OK.
6. For the new template, create a new template definition.

Figure 5-4 Template Definition - Creating an ACK or NAK Workflow Definition

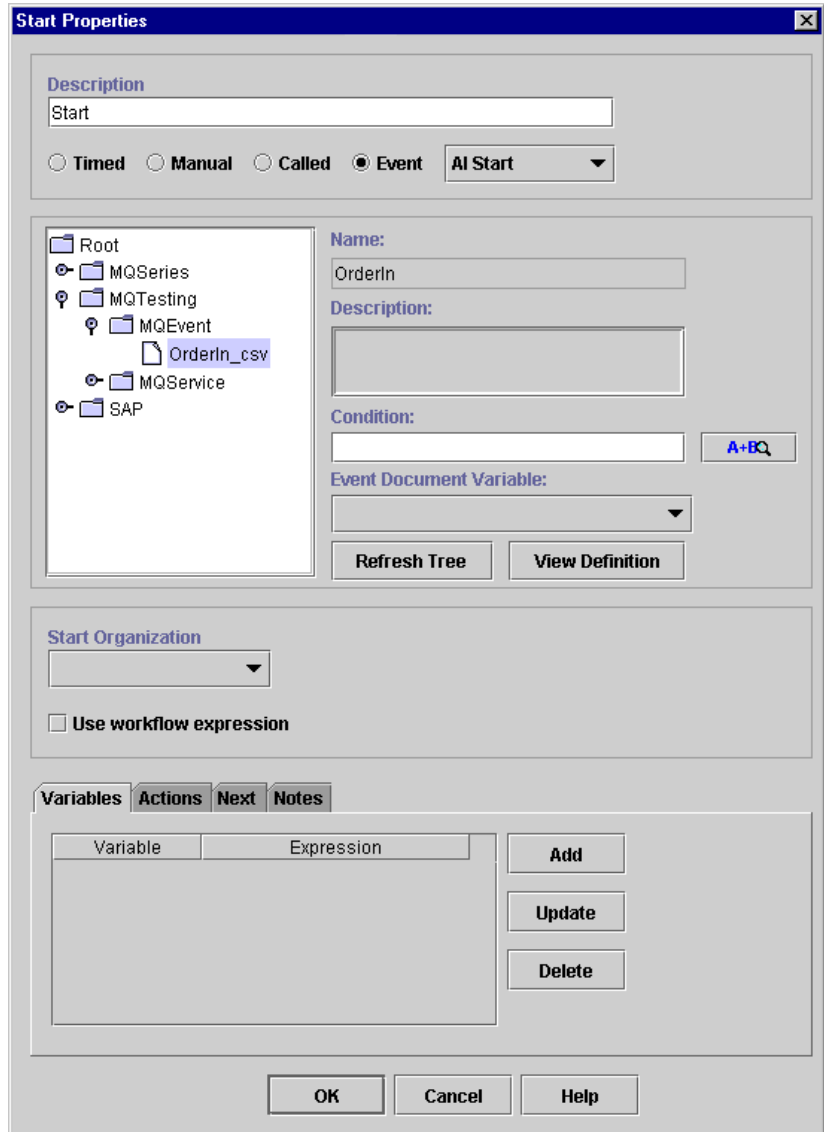
The screenshot shows a dialog box titled "Template Definition SWIFTACK_NAK". It has two tabs: "General" and "Exception Handlers". The "General" tab is active. Inside the dialog, there is a section with the following elements:

- Effective**: A date field showing "Aug 9, 2002" with a dropdown arrow.
- Expiry**: A date field showing "Dec 31, 2002" with a dropdown arrow.
- ☐ **Enable auditing**: An unchecked checkbox.
- Notes**: A text area containing the text "SWIFT Ack or Nak Handling".

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

7. Open the new template definition, select the Start object, and enter the properties.

Figure 5-5 Start Properties Dialog Box - Event Definition



The dialog box is titled "Start Properties" and contains several sections for configuring an event start.

Description: A text field contains the word "Start". Below it are radio buttons for "Timed", "Manual", "Called", and "Event" (which is selected). To the right is a dropdown menu showing "At Start".

Tree View: A folder tree on the left shows the hierarchy: Root > MQSeries > MQTesting > MQEvent > OrderIn_csv (selected). Other folders include MQService and SAP.

Fields: To the right of the tree are fields for "Name:" (containing "OrderIn"), "Description:" (empty), and "Condition:" (empty). There is a search icon "A+BQ" next to the condition field. Below these is an "Event Document Variable:" dropdown menu.

Buttons: "Refresh Tree" and "View Definition" are located below the event document variable field.

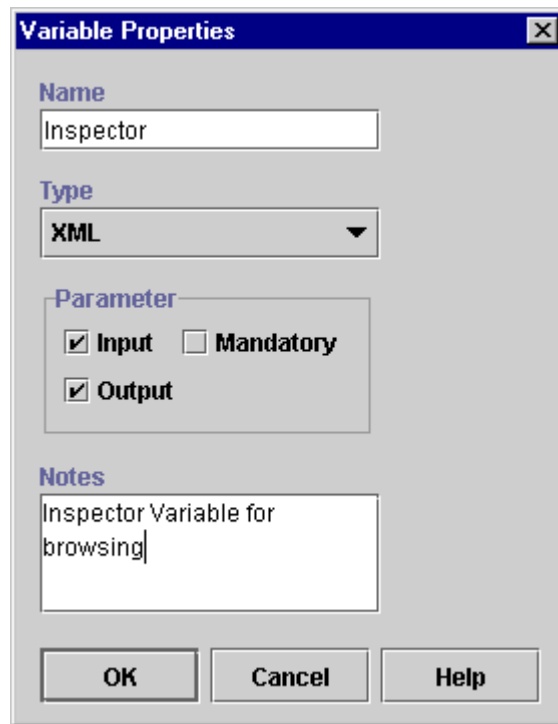
Start Organization: A dropdown menu is present, followed by a checkbox labeled "Use workflow expression".

Variables Section: 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 three buttons: "Add", "Update", and "Delete".

Footer: At the very bottom are "OK", "Cancel", and "Help" buttons.

- a. Select Event →AI Start.
- b. Choose the SWIFT→SWIFTSecurityEvent→ACK_NAK event in the left AI event pane.
- c. Select the Start Organization to be the same as the Template Definition Organization. (See [Figure 5-3.](#))
- d. Add a new Event Document Variable.

Figure 5-6 Variable Properties Dialog Box - Add Start Variable



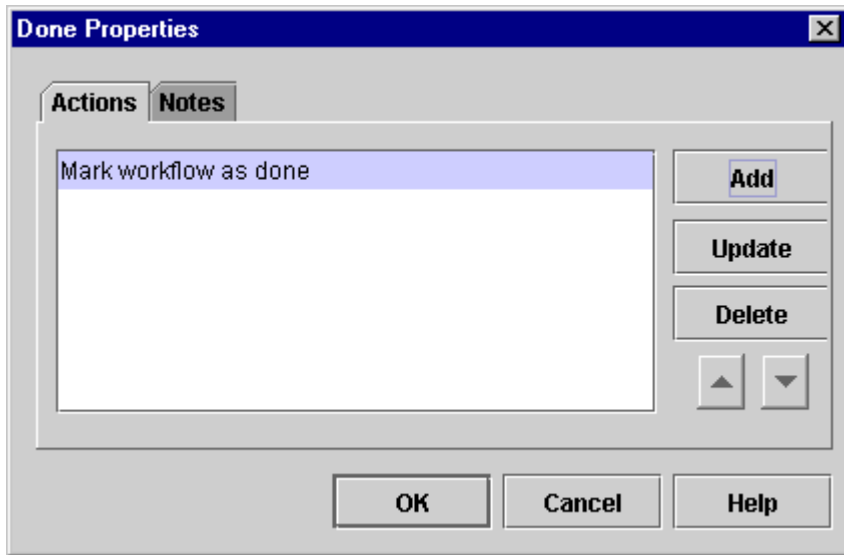
The image shows a 'Variable Properties' dialog box with a blue title bar and a close button. It contains four sections: 'Name' with a text field containing 'Inspector'; 'Type' with a dropdown menu set to 'XML'; 'Parameter' with two checked checkboxes, 'Input' and 'Output', and an unchecked 'Mandatory'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

Section	Field/Control	Value/State
Name	Text Field	Inspector
Type	Dropdown Menu	XML
Parameter	Input	<input checked="" type="checkbox"/>
Parameter	Mandatory	<input type="checkbox"/>
Parameter	Output	<input checked="" type="checkbox"/>
Notes	Text Area	Inspector Variable for browsing

- e. Select Input and Output parameter type.
8. Add an Action to the Done object.
 - a. Select the Done object.
 - b. Click Add an Action.

The Done Properties dialog box appears.

Figure 5-7 Done Properties Dialog Box - Marking Workflow as Done



- c. Click the Actions tab and select Mark Workflow as done.
9. Click OK.
10. Right-click Message Definition in the left pane and select Save.
11. Ensure the workflow is active by selecting the Properties of the workflow definition.

Figure 5-8 Template Definition Properties

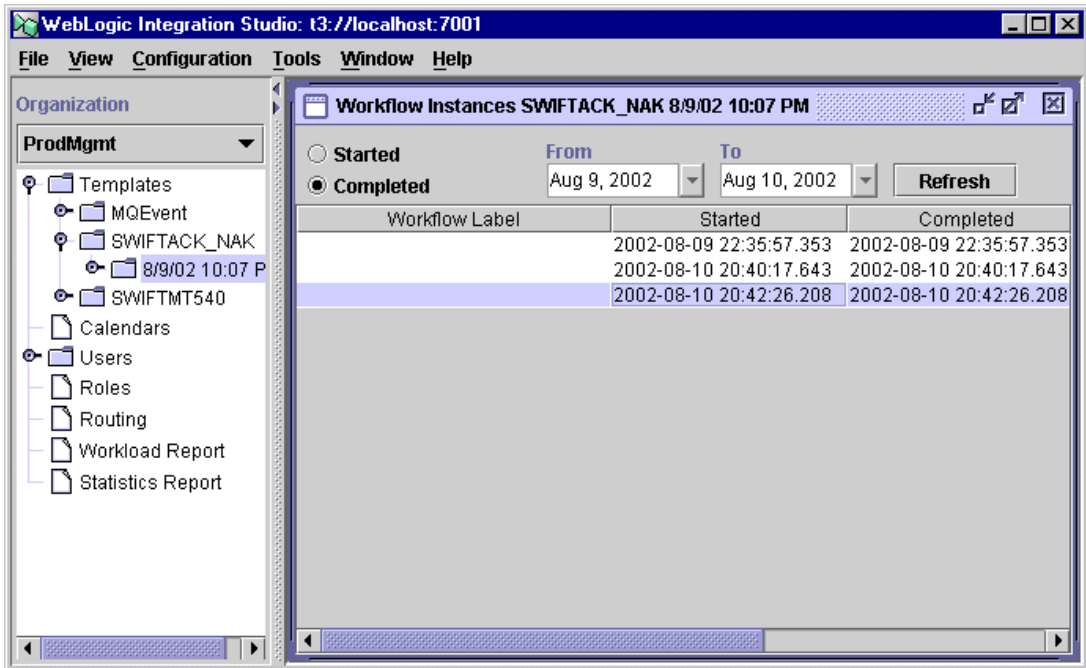
The screenshot shows a dialog box titled "Template Definition SWIFTACK_NAK". It has two tabs: "General" and "Exception Handlers", with "Exception Handlers" currently selected. The dialog contains the following elements:

- Workflow Label:** A text input field with a search icon (magnifying glass) and the text "A+BQ" next to it.
- Active:** A checked checkbox.
- Effective:** A date dropdown menu showing "Aug 9, 2002".
- Expiry:** An unchecked checkbox and a date dropdown menu showing "Dec 31, 2002".
- Enable auditing:** An unchecked checkbox.
- Notes:** A large empty text area.
- Buttons:** "OK", "Cancel", and "Help" buttons at the bottom.

Testing Acknowledgement Message Handling

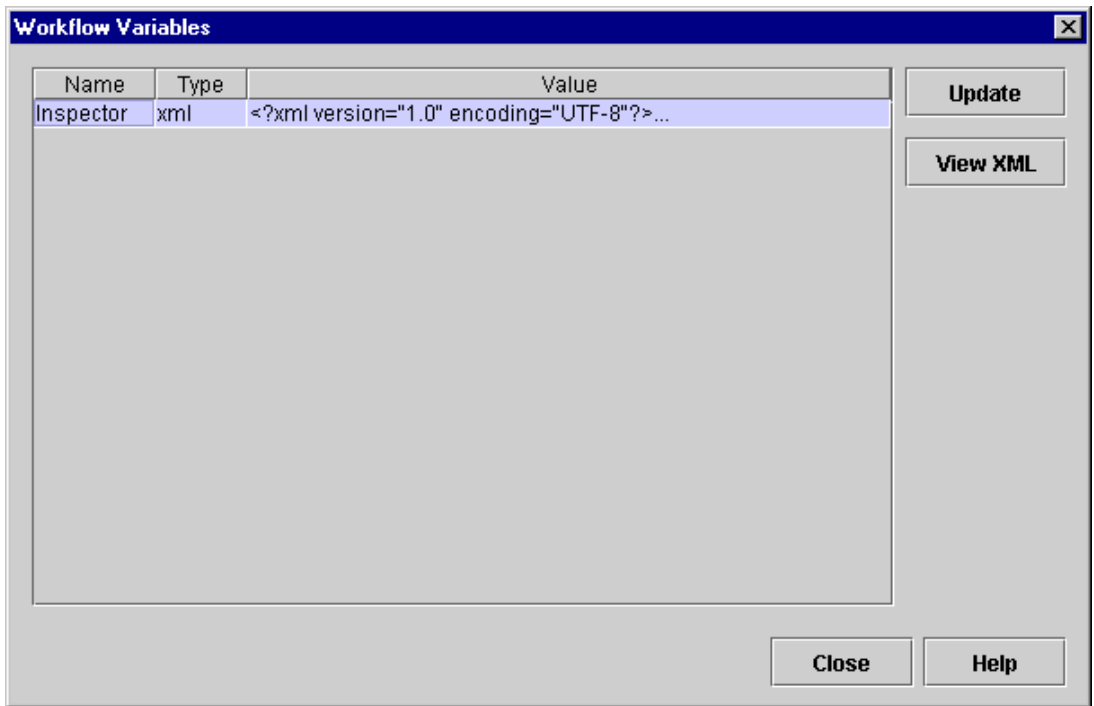
Having created a SWIFT event adapter with two registered events, a SWIFT message event (for example, SWIFT MT540), and a SWIFT ACK or NAK message event, you can view the document as it has been processed through the workflow in the WebLogic Studio console.

Figure 5-9 WebLogic Integration Studio - Template Workflow Instances



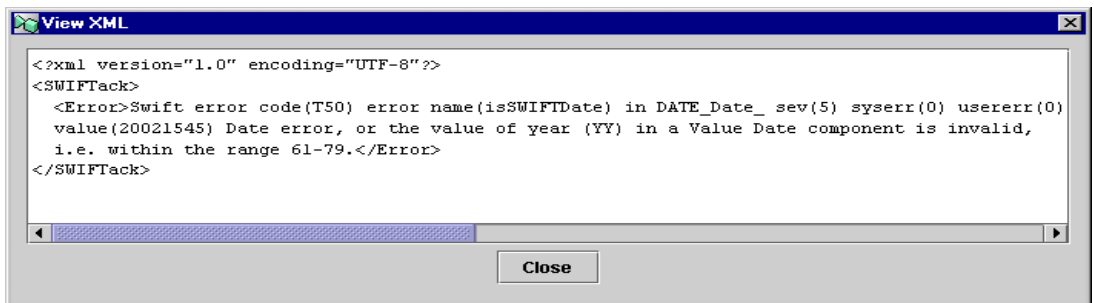
1. Right-click Template and select Instances.
2. Select the instance of interest (that is, the instance generated by the bad message).
3. Right-click and select Workflow Variables:

Figure 5-10 Workflow Variables



4. Click View XML to see the contents of the XML variable "Inspector":

Figure 5-11 Workflow XML Variable



A BEA WebLogic Adapter for ISO15022 Rules System

Document validation enables any document to be validated against sets of rules specified on a per-document basis. The rules are encoded in a rules file that is addressed through the system document dictionary.

Rules apply to document nodes in the XML tree, and (optionally) to their children. Built-in rules can be specified in any combination, and specialized rules can be coded in Java and loaded by the engine, as required.

The following topics discuss how to encode a rules file, how to use the built-in rules, and how to code specialized rules. BEA provides a complete set of built-in rules as needed to validate SWIFT documents. The last section of this document describes how to write rules in Java for special situations. The following topics discuss how to encode a rules file, how to use the built-in rules, and how to code specialized rules:

- [Rules File](#)
- [General Rule Set](#)
- [SWIFT Specific Rule Set](#)
- [Writing Rules in Java](#)
- [Writing Rule Search Routines in Java](#)

Rules File

The rules file is an XML document. One file should exist for each document to be validated. The outer tag should be the document name and under this tag are rule tags, which may be enclosed within USING tags. For example, for a SWIFTMT540, the file (reduced considerably) might look like this:

```
<SWIFTMT540>
  <BASIC>
    <APPID_Application_ID_>F</APPID_Application_ID_>
  ...
  </INFO>
</TRAILER>
</SWIFTMT540>
```

The rules document is an XML document tied into the BEA WebLogic Adapter for SWIFT through `<validation>` tags, which associate one or more rule documents with the specific document entry. The outer tag of the rules document should be explanatory, describing the type of document, but the actual tag is ignored. The rule document itself is a structure containing specific rule applications.

All attributes of rules must be specified in lowercase.

The document entry is the outer tag of the rules file. The name, document, is arbitrary and may be replaced with any meaningful XML-legal name.

Table 5-1 *<document>* entry

Attribute	Use
ackagent	<p>The Java program class to call to construct the acknowledgement. For the BEA WebLogic Adapter for SWIFT, this class is the <code>XDSWIFTACKAgent.class</code>. The actual agent to be used is selected in a search order:</p> <ol style="list-style-type: none">1. The document specification in the repository.2. This attribute of the outer tag of the rule file.3. The listener configuration. <p>As soon as an <code>ackagent</code> is located, it is selected for use, and the search ends.</p>

Table 5-2 <using> entry

Attribute	Use
class	The Java program class contains all <rule>s within the section, unless overridden by a class=attribute in the <rule> entry itself.
other	Any unrecognized attribute is passed to each rule in the rule's attributes. For example, to apply the radix=' , ' attribute to all rules, specify it here rather than on each rule. Rules that do not use the radix attribute would ignore it.

The <rule> tag and method attributes are required. The remaining attributes are rule-specific, and their inclusion is based on the rule itself. The validator uses the required tags to identify the rule in question and to identify the node or nodes of the document to which it applies. Common <rule> tags are:

Table 5-3 <rule> entry

Attribute	Use
tag	Names the right-most parts of the tag to which this rule applies. The rule applies to any node of the document that meets the tag criteria. For example, Document Table Model (DTM) would cause this rule to be applied to every DTM in the incoming document. X.DTM applies to all DTM parts prefixed by X. Tags are case sensitive. If omitted, a stag must be used.
stag	This is a specification subsection tag.
name	The rule's identification, which should be a unique name. This is used in trace messages to specify which rule caused a violation. If omitted, no unique identification can be given.
class	The rule class to which this rule belongs. This corresponds to a Java object class, and each rule is a method of the class. If this is omitted, the class from the enclosing USING tag is used.
method	The specific rule.
usage	Specify usage=M (mandatory) to check that there is a value in the identified node. This check is applied before the actual rule logic is executed.

The rule document is located by the <validation> tag value in the dictionary's system section and is identified with the specific document in its <document> entry. Rules validation is performed for document input (<in_validation>) and/or output (<out_validation>). If several tags are found in the document description, each rules validation is performed in the order in which the tags are found.

A section of the dictionary that illustrates this follows.

```
<system>
  <validation package="SWIFT">
    <name package="SWIFT"
file="templates/SWIFT/rules/SWIFTMT100rules.xml">100RULES</name>
    </validation>
  </system>
  <document> package="SWIFT">SWIFTMT100
    <in_validation>100RULES</in_validation>
  </document>
```

General Rule Set

The engine provides general rules for use by any rule set. The rules are located in `com.ibi.edaqm.XDRuleBase`, which extends `com.ibi.edaqm.XDRuleClass`, the base of all rules. Your own rule class should extend `XDRuleBase` instead of `EXRuleClass`. The general rules include:

- isN
- isR
- isDate
- isTime

isN

isN validates that a node is numeric with an optional leading sign.

```
<rule tag="xx" method="isN" />
```

Table 5-4 isN

Attribute	Meaning
Min	Minimum number of digits required, not including sign. Optional.
Max	Maximum number of digits permitted, not including sign. Optional.

isR

isR validates that a node is numeric with an optional leading sign and a single decimal point.

```
<rule tag="xx" class="XDSWIFTRules" method="isR" />
```

Table 5-5 isR

Attribute	Meaning
Min	Minimum number of digits required, not including sign or radix. Optional.
Max	Maximum number of digits permitted, not including sign or radix. Optional.
Radix	A single character to be used to separate the decimal parts of the real value. The default is a period character. The radix attribute can be taken from the USING entry.

isDate

isDate validates that a node is in CCYYMMDD format.

```
<rule tag="xx" class="XDSWIFTRules" method="isDate" />
```

Table 5-6 isDate

Attribute	Meaning
Min	Minimum number of positions required. If omitted, 8 is assumed.
Max	Maximum positions permitted. If omitted, 8 is assumed.

isTime

isTime validates that a node is in HHMM[SS] format.

```
<rule tag="xx" method="isTime" />
```

Table 5-7 isTime

Attribute	Meaning
None	None.

SWIFT Specific Rule Set

The SWIFT specific rules available include:

- isValidReference
- isValidISIN
- isNotPresent
- isValidMultiLine
- isSWIFTReal
- isSWIFTDate
- isValidSWIFTString

- isSWIFTTime
- isValidMessageType
- checkValue
- checkCD
- checkRepetitive
- checkNodes
- checkChildSequence
- checkAddition
- checkRelation
- checkSegment

isValidReference

isValidReference validates the value with the following validations:

- The first character should not be /.
- The last character should not be /.
- At any place no two / should come together.

```
<rule tag="(node to check)" method="isValidReference"
errorcode="(error code)"/>
```

isValidISIN

isValidISIN validates the value with the following validations:

- The first four characters should be ISIN.
- The fifth character should be a space.

- The maximum characters should be 17 including the ISIN and the space which follows it.

```
<rule tag="(node to check)" method="isValidISIN" errorcode="(error code)"/>
```

Warning: The format of the ISIN given in the input is valid. The validity of the ISIN can be checked in the SWIFT ISIN directory.

isNotPresent

isNotPresent validates whether or not the value is present.

```
<rule tag="(node to check)" method="isNotPresent" errorcode="(error code)"/>
```

isValidMultiLine

isValidMultiLine validates whether or not the value is present.

The value should be alphanumeric.

The line ends with a carriage return and a new line character.

```
<rule tag="(node to check)" method="isMultiLine" line="3" min="2" max="10" errorcode="(error code)"/>
```

Table 5-8 isValidMultiLine

Attribute	Meaning
Line	Number of valid lines. The minimum number of lines is one (1).
Min	Minimum number of characters.
Max	Maximum number of characters.
Errorcode	The error code for the rule.

isSWIFTReal

isSWIFTReal validates the value with the following validations:

- The value is checked for real numbers with at least one character before the decimal point (the decimal point is “,”).
- The value is checked for having a mandatory decimal point.

Note: The decimal comma is included in the maximum length.

```
<rule tag="(node to check)" method="isSWIFTReal" min="2" max="10"
errorcode="(error code)"/>
```

Table 5-9 isSWIFTReal

Attribute	Meaning
Min	Minimum number of characters.
Max	Maximum number of characters.
Errorcode	The error code for the rule.

isSWIFTDate

isSWIFTDate validates the date according to the format specified in the attribute. The following are valid formats:

- date1 : MMDD.
- date2 : YYMMDD.
- date3 : YYMM.
- date4 : YYYYMMDD.
- date5 : date4 + value date (the year should be between 1980 and 2060).

```
<rule tag="node to check" method="isSWIFTDate" format="date1" />
```

Table 5-10 isSWIFTDate

Attribute	Meaning
Format	The format of the date to check against.
Errorcode	The error code for the rule.

IsValidSWIFTString

IsValidSWIFTString validates the value according to the format specified.

The valid values of the format are:

- x—The S.W.I.F.T X character list.
- y—The S.W.I.F.T Y character list.
- z—The S.W.I.F.T Z character list.
- c—Alphanumeric capital letters and digits only.
- a—Alphabetic capital letters only.
- h—Hexadecimal characters only.
- e—Blank spaces.

Exceptions to this rule are:

- A value cannot have blank spaces alone.
- A value cannot have CrLf characters alone.

IsValidSWIFTString validates each and every digit and/or character against the standard SWIFT recognized X character list.

Table 5-11 SWIFT X Character Set

Character Set
A to Z (uppercase)
a to z (lowercase)
0 to 9
/ (forward slash), - (minus sign), ? (question mark), : (colon), ((left parenthesis),) (right parenthesis), . (point), , (comma), ' (right single quote), + (plus sign), SPACE, CrLf (line feed, new line, and carriage return characters)

Table 5-12 SWIFT Y Character Set

Character Set
A to Z (uppercase)
0 to 9
/ (forward slash), - (minus sign), ? (question mark), : (colon), ((left parenthesis),) (right parenthesis), . (point), (comma), ' (right single quote), + (plus sign), SPACE, = (equal to)
! (exclamation mark), " (right quotes), % (percentage), & (ampersand), * (asterisk), ; (semi-colon), < (left V bracket), > (right V bracket)

Table 5-13 SWIFT Z Character Set

Character Set
A to Z (uppercase)
a to z (lowercase)
0 to 9
/ (forward slash), - (minus sign), ? (question mark), : (colon), ((left parenthesis),) (right parenthesis), . (point), , (comma), ' (right single quote), + (plus sign), SPACE, Cr Lf (line feed, new line, and carriage return), = (equal to), @ ,#, { (left brace)

Table 5-13 SWIFT Z Character Set (Continued)

Character Set
! (exclamation mark),” (right quotes), % (percentage), & (ampersand), * (asterisk), ;(semi-colon), < (left V bracket), >(right V bracket)

Hexadecimal Representation of SWIFT Character Set

SWIFT characters can be comprised of hexadecimal characters. These representations are different from regular IBM hexadecimal representations of characters.

Figure 5-12 Hexadecimal Representation of SWIFT Character Set

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0					Sp	&	-						{			0
1							/		a	j			A	J		1
2									b	k	s		B	K	S	2
3									c	l	t		C	L	T	3
4									d	m	u		D	M	U	4
5			Lf						e	n	v		E	N	V	5
6									f	o	w		F	O	W	6
7									g	p	x		G	P	X	7
8									h	q	y		H	Q	Y	8
9									i	r	z		I	R	Z	9
A								:								
B					.		,	#								
C					<	*	%	@								
D	Cr				()		'								
E					+	;	>	=								
F					!		?	"								

For example, an Lf character can be depicted as 25, an & can be depicted as 50, and an ! can be depicted as 4f.

```
<rule tag="(node to check)" method="isValidSWIFTString" type="x"
errorcode="(error code)"/>
```

Table 5-14 isValidSWIFTString

Attribute	Meaning
Type	The type of string format to check against.
Min	Minimum number of characters.
Max	Maximum number of characters.
Errorcode	The error code for the rule.

isSWIFTTime

isSWIFTTime validates the value according to the format specified.

The valid values of the format are:

- time1: HHMM.
- time2: HHMMSS.
- time3: **HHMMSS**sss (where sss stands for microseconds).

```
<rule tag="(node to check)" method="isValidSWIFTTime"
format="time1" errorcode="(error code)"/>
```

Table 5-15 isSWIFTTime

Attribute	Meaning
Format	The type of date format to check against.
Errorcode	The error code for the rule.

isValidMessageType

isValidMessageType validates the message type value.

The message type value should be greater than 100 and less than 999.

There is an exception if the message is one of the following sets:

101,102,204,206,207,256,303,304,405,416,503,504,505,506,507,527,569, or 575.

A warning message indicating that sending or receiving one of the above messages requires a MUG registration, and that special validation is required as per individual MUG standards, is sent to the console and/or logged in the trace file.

```
<rule tag="(node to check)" method="isValidMessageType"
errorcode="(error code)"/>
```

Table 5-16 isValidMessageType

Attribute	Meaning
Errorcode	The error code for the rule.

checkValue

checkValue validates the elements or components of a segment for presence according to a pattern. The patterns validate the code in the independent variable. The tag= and tagset= attribute can be used to locate the section to which the rule applies.

Table 5-17 CheckValue

Attribute	Meaning
Tagset	List of nodes to check against.
Code	The expression used to validate the elements.
Errorcode	The error code for the rule.

Case 1

```
<rule tag="Parent of A and B" method="checkValue" tagset="A,B"
code="1=@reasonList/2=@statusList" errorcode="(error code)"/>
```

If the Value of A is one of the code set 'reasonList,' then the value of B should be one of the code set of statusList.

Case 2 `<rule tag="Parent of A and B" method="checkValue" tagset="A,B" code="1=x;y;z/2=a;b;c" errorcode="(error code)" />`

If the Value of A is one of the set {x,y,z}, then Value of B should be one of the set { a, b, c}.

Case 3 `<rule tag="Parent of A and B" method="checkValue" tagset="A,B" code="1!/?/2=?" errorcode="(error code)" />`

If element A is not present, element B should be present.

Case 4 `<rule tag="Parent of A,B" method="checkValue" tagset="A,B" code="1=?/2=?,1!/?/2!?" errorcode="(error code)" />`

Either A and B should be present, or A and B should not be present.

Case 5 `<rule tag="Parent of A,B, C" method="checkValue" tagset="A,B,C" code="1=?/2!/?+3!?,1!/?/2=?+3=?" errorcode="(error code)" />`

Either element A or (B and C) must be present.

Case 6 `<rule tag = "Parent of x,y,z,A,B" method="checkValue" tagset="x,y,z,A,B" code="1=? | 2=? | 3=?/4=?+5=? errorcode="(error code)" />`

If any of the elements x, y, and z are present, then element A and element B must be present.

Case 7 `<rule tag = "Parent of A, X, Y, Z, x, y, z, ..." method = "checkValue" tagset = "A, X, Y, Z, x, y, z, ..." Code = "1=XXX + (2=? | 3=? | 4=?...) / 5=? | 6=? | 7=?... errorcode="(error code)" />`

If element A contains the word XXX and any of the elements {X,Y,Z,X1,Y1,Z1,...} are present, then any of the elements {x,y,z,x1,y1,z1....} should be present.

checkCD

checkCD validates that a node has appropriate sub nodes. The valid codes and definitions of the codes are as follows:

Table 5-18 checkCD Valid Codes

Condition Code	Meaning	Definition
R	Required	At least one of the elements in the condition must be present.
E	Exclusion	Not more than one of the elements specified in the condition can be present.
G	Repetitive Sequence Related	The first element must be present if there are no repetitive sequences of the second or third element.
F	Repetitive Sequence Related	The first element must be present if there are repetitive sequences of the second element.
V	Multiple Occurrences Related	If the first element is present and the second element is present at least once, then the value of all the occurrences of the second element should be equal to the first element.
M	Mandatory	One of the children of the specified elements must contain a value.

Table 5-19 checkCD

Attribute	Meaning
Tagset	List of nodes to check against.
Cd	The expression used to validate the elements.
Errorcode	The error code for the rule.

Case 1 `<rule tag="Parent of A and B" method="checkCD" tagset="A,B" cd="E0102" errorcode="(error code)"/>`

Mutually exclusive A and B.

Case 2 `<rule tag="Parent of A,B" method="checkCD" tagset="A,B" cd="R0102" errorcode="(error code)"/>`

Either element A or element B or both must be present.

Case 3 `<rule tag="Parent of A,B, C" method="checkCD" tagset="A,B,C" cd="G010203" errorcode="(error code)"/>`

Element A must be present if there are no repetitive sequences B or C (this is applicable to MT 573).

Case 4 `<rule tag="Parent of A,B" method="checkCD" tagset="A,B" cd="F0102" errorcode="(error code)"/>`

Element A must be present if there are repetitive sequences of element B (two or more times).

Case 5 `<rule tag="Parent of A,B" method="checkCD" tagset="A,B" cd="V0102" errorcode="(error code)"/>`

If element A is present and there is at least one element B, then the value of all occurrences of element B should be equal to the value of element A.

Case 6 `<rule tag="Parent of A,B" method="checkCD" tagset="A,B" cd="M0102" errorcode="(error code)"/>`

One of the children from element A and element B must contain a value.

checkRepetitive

`checkRepetitive` validates that the value of the specified element should be the same in all occurrences if the specified element is used or present repetitively.

```
<rule tag="(Root of the document)" search="A" errorcode="(error code)"/>
```

Table 5-20 checkRepetitive

Attribute	Meaning
Search	The node to search for and check.
Errorcode	The error code for the rule.

Note: A is the name of the actual node (not a fully qualified name). It is specified the root of the document, instead of the parent of A, because this rule searches the entire document for A.

checkNodes

`checkNodes` validates either one or more of elements from a set A, A1, ... of nodes must be present or one or more of the set B, B1, ... of nodes must be present, but not elements from both sets. For example, {A, A1, A2, ..., } and {B, B1, B2, ...} are two sets of nodes defined. Only nodes from one of the sets can exist. If node A and node B exists, then the rule fails.

```
<rule tag="Parent of A,A1,...B,B1,..." method="checkNodes"
tagset="A,A1,A2,A3,...;B,B1,B2,B3....." errorcode="(error code)" />
```

Table 5-21 checkNodes

Attribute	Meaning
Tagset	The two sets of nodes to check separated by a semicolon (;).
Errorcode	The error code for the rule.

checkChildSequence

`checkChildSequence` checks for a specified node for its occurrences, and the presence of other nodes is based upon it.

Table 5-22 checkChildSequence

Attribute	Meaning
Start	Node to check if it is repetitive.
Pattern	The expression used to validate the elements.
Errorcode	The error code for the rule.

Case 1 `<rule tag="Parent of A" method="checkChildSequence" start = A
pattern= B errorcode=" (error code)" />`

If the tag A is repetitive and is present two or more times, then tag B should be one of the children of tag A.

Case 2 `<rule tag="Parent of A" method="checkChildSequence" start = "A"
pattern= "B1,B2,B3..." errorcode=" (error code)" />`

If the tag A is repetitive and is present two or more times, then one of the tags {B1,B2,B3,...} must be present.

checkAddition

checkAddition validates the value of a specified element to be equal to the sum of all values of another element.

Table 5-23 checkAddition

Attribute	Meaning
Tagset	List of nodes to check.
Errorcode	The error code for the rule.

Case 1 `<rule tag="Parent of A , B, C" method="checkAddition"
tagset="A,B,C" errorcode=" (error code)" />`

The value of element A should be equal to the sum of all values of element B (they are repetitive optionally) or the sum of all values of element C (they are repetitive optionally).

The validation applies only if the element A exists in the incoming SWIFT message. If the element A exists, then at least one of the elements of B should exist or at least one of the elements of C should exist.

Case 2 `<rule tag="Parent of A , B" method="checkAddition" tagset="A,B,C"
errorcode=" (error code)" />`

The value of element A should be equal to the sum of all values of element B (they are repetitive optionally).

The validation applies only if the element A exists in the incoming message. If the element A exists, then at least one element B should exist.

checkRelation

checkRelation validates whether element A or one or more of the set {x,y,z,...} is present, then element B should be present and must be succeeding all occurrences of A or one or more of the set {x,y,z,...}. The converse is also true.

```
<rule tag="Parent of A,B,x,y,z,...," method="checkRelation"
tagset="A" taglist="x,y,z" find="B" errorcode="(error code)" />
```

Table 5-24 checkRelation

Attribute	Meaning
Tagset	Node to check.
Taglist	List of nodes to check.
Find	Node to find to see if it should be present.
Errorcode	The error code for the rule.

checkSegment

Segment D is mandatory when in any occurrence of segment C, sub-segment C1 is present, and the sub-segment C1a is not present.

Note: C1a is the child of C1. When specified in the rule, specify the node with its full name.

```
<rule tag="Parent of C,C1,C1a,D" method="checkSegment" parent="C"
subseq="C.C1" child="C.C1.C1a" check="D" errorcode="(error code)"
/>
```

Table 5-25 checkSegment

Attribute	Meaning
Parent	The parent node.
Subseq	The sub-segment node.
Child	The child node.
Check	The node to check if present.
Errorcode	The error code for the rule.

Writing Rules in Java

Rules can be written in Java, loaded by the system at startup, and applied by specification in a rule. A rule class extends `XDRuleClass` and can make use of any of its services. Each public method in the rule class that meets the rule signature can be applied by name as a rule. The rule methods can make use of service methods in the parental `XDRuleClass`.

In this example, a node is checked to determine whether its value is the word identified by the `value=` attribute. If not, it is an error.

The following parameters are passed:

Table 5-26 Parameters Passed In Method That Checks Node

Parameter	Meaning
Node	The node identified by the tag attribute in the rule. The rule method will be called once for each node that matches the tag specification.

Table 5-26 Parameters Passed In Method That Checks Node (Continued)

Value	The data value of the addressed node. This differs from the <code>node.getValue()</code> return if the tag contained a subfield address (for example, <code>tag=x:2</code>).
Attributes	A <code>HashMap</code> of rule attributes. The rule method can check for any attributes that it requires. A <code>HashMap</code> is a fast implementation of a <code>Hashtable</code> that does not serialize.

Listing A-3 Node Checking Example

```
import java.util.*;
import com.ibi.edaqm.*;
public class XDMyRules extends XDRuleClass
{
    public XDMyRules()
    {
    }
    public void specialRule(XDNode node, String value,
                          HashMap attributes)
        throws XDEException
    {
        trace(XD.TRACE_DEBUG, "specialRule called with parms: " +
            node.getFullName() + ", " + attributes.toString());
        String testValue = (String)attributes.get("value");
        if (value.equals(testValue) )
        {
            node.setAssociatedVector(new XDEDIError(4, 0,
            error,"explanation"));
            throw new XDEException(XD.RULE, XD.RULE_VIOLATION, "node
            value
            "+value+" is not 'Value'");
        }
    }
}
```

Rule violations should throw an `XDEException` describing the violation.

The parental class provides a group of services to assist in preparing rules:

Table 1-27 Services for Preparing Rules

Method	Purpose
Boolean is YYYYMMDD (string date).	Validates that a date is formatted correctly.
Boolean is InList (string list, string value).	The value must be in the list.
Void trace (int level, string msg).	The text of the message is written to the system trace file. The level should be one of the following values: <ul style="list-style-type: none"> ■ XD.TRACE_DEBUG ■ XD.TRACE_ERROR ■ XD.TRACE_ALL

Rules can also use all methods in `XDNode` to address the values in the passed node and the tree in general.

Rule violations must be returned as `XDExceptions` of class `XD.RULE`. Two causes are available, `XD.RULE_SYNTAX` if the rule is in error, and `XD.RULE_VIOLATION` if the data violates the rule. Syntax errors cause the document to be aborted, as it is presumed that rules should have been debugged. Violations should be posted to the node by the rule, and the engine continues to process the document. Violations are traced by the engine and affect the later acknowledgement generation.

The error itself is posted to the node by the standard `XDNode` service `setAssociatedVector(Object o)` which records an object with the node. The special `EDIError` object contains the elements:

Table 1-28 Elements in the EDIError Object

Element	Description
Class	Class of the error. Should be 4 for a syntax error, resulting in an AK4.
Reserved	Must be 0.
Error code	Code to be returned in the ack AKx (997).
Explanation	A string explaining the error, for tracing use.

Writing Rule Search Routines in Java

Short lists can be searched by built-in rule engine code. Longer lists, in which the values in the list are obtained not from the attribute directly, but instead from an external source, require a rule list searcher tailored to the source. Lists might be obtained from:

- A simple file.
- A database with values loaded at startup.
- A database with an access at each search request.

Each list might require its own search logic, tailored to the source and format of the list itself. To accommodate this, the rule engine allows list-specific search routines to be developed and added to the system. These routines are loaded at system initialization and terminated at system shutdown. Each must offer a search method that determines whether the passed value is valid.

Search routines must extend the `XDRuleList` class that is part of the `edagm` package: `com.ibi.edagm.XDRuleClass`. The routine must offer these methods in the manner common to all XD extensions:

- `init(String args)` is called once at system initialization.
- `term()` is called once at system termination. It is not guaranteed that it is called.
- `search(String value)` is called when the rule is executed.

The Rule List search code is identified in the `<preload>` section of the `<system>` area of the dictionary. The Preloads console page manages this section.

```
<preload>
  <name file="RuleFileList(c:\ziplist.txt)"
  comment="validates zip codes">ziplist</name>
</preload>
```

The `<rule>` tag specifies that a rule can be written:

```
<rule tag="xxx" code="@ziplist" method="checklist"/>
```

that names the preloaded routine. This routine might load a list from a text file. A simplified example procedure to load a file containing codes follows:

Listing A-4 Loading a File Example

```
import com.ibi.edaqm.*;
import java.util.*;
import java.io.*;
/**
 * A rule list handler is a routine called to enable users search lists during
 * execution
 * or the checkList rule. checkList() is a generally available rule to test whether the
 * contents of a document field are valid. The rule list handler is invoked when
 * the code= attribute indicates the name of a coder routine rather than a simple
 * list.<P>
 * For example, <I>code="@list1"</I> will cause the search routine of the list1 class
 * to
 * be invoked.<P>
 * The file read by this procedure consists of tokens separated by new line, white
 * space or commas.
 */
public class XDRuleListFile extends XDRuleList
{
    String[] list;
    ArrayList al = new ArrayList(127);
    public XDRuleListFile()
    {
    }
    /**
     * The init method is called when a rule is loaded. It can perform any necessary
     * initialization, and can store any persistent information in the object store.
     *
     * @param parms Array of parameter string passed within the start command
     * init-name(parms).
     */
    public void init(String[] parms) throws XDException
    {
        if (parms == null)
        {
            throw new XDException(XD.RULE, XD.RULE_SYNTAX, "no parms sent to " +
name);
        }
        try
        {
            File f = new File(parms[0]);
            FileInputStream fs = new FileInputStream(f);
            long len = f.length();
            byte[] b = new byte[(int)len];
            fs.read(b);
            fs.close();
            String data = new String(b);
            StringTokenizer st = new StringTokenizer(data, " " +
XD.NEWLINE);
```

```
        while (st.hasMoreTokens())
        {
            String part = st.nextToken();
            al.add(part);
        }
    }
    catch (FileNotFoundException e)
    {
        throw new XDException(XD.RULE, XD.RULE_SYNTAX, "list file
"+parms[0] + " not found");
    }
    catch (IOException eio)
    {
        throw new XDException(XD.RULE, XD.RULE_SYNTAX, eio.toString());
    }
}
/**
 * The term() method is called when the worker is terminated. It is NOT
guaranteed
 * to be call, and applications should not rely upon this method to update
data bases or
 * perform other critical operations.
 */
public void term()
{
}

/**
 * Search the given value to determine whether it is in the list.
 *
 * @param value String to test against the list
 * @return true if found, false otherwise
 */
public boolean search(String value)
{
    return al.contains(value);
}
}
```

B Linking the BEA WebLogic Adapter for ISO15022 to a SWIFT Network

This section describes the connecting of business applications to SWIFTAlliance. It includes the following topics:

- [Batch File Transfer – FILE and FTP](#)
- [Application Server – CAS MF](#)
- [Interactive – MQ Series](#)

SWIFT allows the connecting of business applications to SWIFTAlliance which can be done in different ways:

- **Manual File Transfer:** S.W.I.F.T. messages are exchanged between the business application and SWIFTAlliance in batch files, *and* with operator intervention on (either and/or both) the business application and the SWIFTAlliance.
- **Automated File Transfer:** S.W.I.F.T. messages are exchanged between the business application and SWIFTAlliance in batch files, *without operator intervention*, that is, the file transfer operation is automated on (either or/and both) SWIFTAlliance and Business Application.

- **Interactive:** Unlike file transfer operation mode, S.W.I.F.T. messages are exchanged real time (via a conversational protocol) between the business application and SWIFTAlliance on an individual basis and *without operator intervention* on both SWIFTAlliance and Business Application.

The BEA WebLogic Adapter for SWIFT supports all three options when used in conjunction with other BEA adapters such as FTP, File, TCP/IP, and MQ Series. The following table summarizes the connectivity options:

Table 1-29 Connectivity Options for BEA WebLogic Adapters

Mode	BEA Adapter	3 rd Party Software
File Transfer	BEA WebLogic Adapter for SWIFT BEA WebLogic Adapter for File	None specific but could be application – for example, MERVA or BESS
Application Server	BEA WebLogic Adapter for SWIFT BEA TCP/IP Adapter	'AI MXA TCP-IP' CASmf Application Server
Interactive	BEA WebLogic Adapter for SWIFT BEA WebLogic Adapter for MQSeries	MQSA SWIFT Alliance Toolkit Runtime

Batch File Transfer – FILE and FTP

The File Transfer method permits batch file transfer with message partners. This method permits both automated and manual invocation of communication sessions, either with or without the use of parameter files. For each message format the communication media may be diskettes or files, that is, read or write a batch message file from, or to a directory in a local or remote file system.

The following message file formats are supported:

- **DOS-PCC** is used for batch input and output of messages. The DOS-PCC connection method permits you to read or write an ST200 DOS message file.
- **RJE** (Remote Job Entry) is used for batch input and output of messages in ST200 RJE format.
- **MERVA/2** batch transfer (from/to mainframes) in IBM MERVA/2 format.
- **CAS** (Common Application Server) format.

Application Server – CAS MF

The SWIFT Common Application Server (CAS) defines how data can be exchanged between SWIFTAlliance and financial applications via a conversational protocol. The specifications are common to both SWIFTAlliance and ST400, thus allowing smooth ST400 migration to SWIFTAlliance.

CAS supports TCP/IP and SNA LU6.2 as a networking protocol. Application developers need not know the CAS protocol. The CASmf software package provides APIs to the financial application developers. It hides all communication and data formatting aspects enabling developers to concentrate on the application functions. APIs exist to open/close/abort a session and send/receive data.

CASmf is available on the following platforms:

- AIX
- HP/UX
- SunOS
- Windows NT
- AS400
- Open VMS for VAX and Alpha

CASmf uses TCP/IP as the communication protocol. It can run on the same system as SWIFTAlliance. CASmf can handle several simultaneous application sessions. The BEA WebLogic Adapter for SWIFT can use TCP/IP Adapter for processing events and services between WebLogic Integration and SWIFT CASmf applications. The option 'AI MXA TCP-IP' must be licensed on SWIFTAlliance.

Interactive – MQ Series

The MQSeries Interface for SWIFTAlliance (MQSA) software provides a reliable communication between financial applications and SWIFTAlliance through IBM MQSeries. It enables S.W.I.F.T. messages exchange between SWIFTAlliance and financial applications. The MQSA software is based on the SWIFTAlliance Development Toolkit referenced as ADK in this document. It uses ADK functions to communicate with SWIFTAlliance and MQSeries functions to access message queuing services.

The MQSeries Interface for SWIFTAlliance is available for SWIFTAlliance Access running on Windows NT and UNIX. IBM MQSeries messaging software enables business applications to exchange information across different operating system platforms in a way that is straightforward and easy for programmers to implement.

MQSeries is available on different platforms (Windows NT, UNIX, OS/400, MVS/ESA, and so forth). The applications are shielded from the mechanics of the underlying communications. With MQSeries, the exchange of messages between the sending and receiving program is time independent. This means that the sending and receiving applications are de-coupled so that the sender can continue processing without having to wait for the receiver to acknowledge the receipt of the message.

The MQSA software enables fast integration of user applications with SWIFTAlliance Access. It is composed of two ADK components, and they must be installed as any other ADK component. The components must be registered in SWIFTAlliance. The purpose of the registration is to make the component known to SWIFTAlliance. The registration adds component data to the SWIFTAlliance database. The MQSA software is limited to the exchange of S.W.I.F.T. messages, S.W.I.F.T. ack/nacks and recording of events in the SWIFTAlliance journal. It can handle multiple MQSeries queues for the connection with the user application(s).

The BEA WebLogic Adapter for SWIFT and BEA WebLogic Adapter for MQSeries provide connectivity to SWIFT network using SWIFT Alliance MQSA. The SWIFTAlliance Toolkit run-time license is required to run the MQSA software.

