



BEA WebLogic Adapter for TIBCO™

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 TIBCO User Guide

Part Number	Date
N/A	November 2002

Table of Contents

About This Document

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

1. Introducing the BEA WebLogic Adapter for TIBCO

TIBCO Rendezvous.....	1-1
The BEA WebLogic Adapter for TIBCO	1-3
How the BEA WebLogic Adapter for TIBCO Works	1-4

2. Metadata, Schemas, and Repositories

Understanding Metadata.....	2-2
Schemas and Repositories	2-5
Naming a Schema Repository	2-6
The Repository Manifest	2-7
Creating a Repository Manifest.....	2-8
Creating a Schema	2-9
Storing Directory and Template Files for Transformations	2-13
Samples File	2-13

3. Creating and Configuring an Event Adapter

Creating an Application View Folder.....	3-1
Creating an Event Adapter Application View.....	3-3
Configuring an Event Adapter Application View	3-8
Testing Event Adapter Application Views Using the Application View Console..	3-15

Testing Event Adapter Application Views Using WebLogic Integration Studio...	3-18
--	------

4. Creating and Configuring a Service Adapter

Creating a Service Adapter Application View	4-1
Configuring a Service Adapter Application View.....	4-4
Testing the BEA Service Adapter for TIBCO	4-10
Testing Service Adapter Application Views Using WebLogic Integration Studio	4-15

5. Transforming Document Formats

Kinds of Transformation	5-1
Non-XML to XML Transformation	5-2
XML to XML Transformations.....	5-4
XML to Non-XML	5-6
Creating and Testing Transformations	5-8
Creating MFL Transformations.....	5-8
Testing MFL Transformations	5-12

6. Creating Schema Repositories

Introduction to Schemas and Repositories	6-1
Naming a Schema Repository	6-2
The Repository Manifest	6-3
Creating a Repository Manifest.....	6-4
Creating a Schema	6-5

A. Troubleshooting

Error Messages	A-1
----------------------	-----

B. Sample Integration With a SWIFT Message

The TIBCO-Swift Integration Scenario	B-2
Creating an Event Adapter Application View	B-3
Configuring an Event Adapter Application View	B-7
Validating Deployment Using WebLogic Integration Studio	B-12
Publishing the Message Using RTTM.....	B-15

About This Document

This document explains how to install, configure, and deploy the BEA WebLogic Adapter for TIBCO to develop online connections to TIBCO applications.

This document is organized as follows:

- [Chapter 1, “Introducing the BEA WebLogic Adapter for TIBCO,”](#) provides an overview of the BEA WebLogic Adapter for TIBCO.
- [Chapter 2, “Metadata, Schemas, and Repositories,”](#) describes metadata, how to name a schema repository and the schema manifest, how to create a schema, how to store directory and template files for transformations.
- [Chapter 3, “Creating and Configuring an Event Adapter,”](#) describes how to create, configure, and test event adapter application views.
- [Chapter 4, “Creating and Configuring a Service Adapter,”](#) describes how to create, configure, and test a service adapter application view.
- [Chapter 5, “Transforming Document Formats,”](#) describes how the BEA WebLogic Adapter for TIBCO supports inbound transformations for the event adapter and outbound transformations for the service adapter.
- [Chapter 6, “Creating Schema Repositories,”](#) explains how to name a schema repository, create a repository manifest, and create a schema.
- [Appendix A, “Troubleshooting,”](#) lists error messages you may encounter while using the BEA WebLogic Adapter for TIBCO.
- [Appendix B, “Sample Integration With a SWIFT Message,”](#) illustrates how to send a non-XML SWIFT message between TIBCO Rendezvous and WebLogic Integration.

What You Need to Know

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

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

If you have any questions about this version of the BEA WebLogic Adapter for TIBCO, or if you have problems installing and running the BEA WebLogic Adapter for TIBCO, 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 TIBCO

This section introduces the BEA WebLogic Adapter for TIBCO, providing an overview of its key features and of how you can use it to integrate TIBCO Rendezvous messages with WebLogic Integration. This section includes the following topics:

- [TIBCO Rendezvous](#)
- [The BEA WebLogic Adapter for TIBCO](#)
- [How the BEA WebLogic Adapter for TIBCO Works](#)

TIBCO Rendezvous

TIBCO Rendezvous is the messaging system that is the foundation of TIBCO ActiveEnterprise. Rendezvous delivers true real-time publish or subscribe and request or reply messaging. It also supports qualities of service ranging from lightweight informational messages to certify and transactional delivery.

Rendezvous utilizes a distributed architecture to eliminate bottlenecks and single points of failure. Applications can select from several qualities of service including reliable and certified and transactional, as appropriate for each interaction. Messaging can be request or reply, publish or subscribe, synchronous or asynchronous, and locally delivered or sent using a WAN or the Internet. Rendezvous messages are self-describing and platform independent, with a user-extensible type system that provides support for data formats such as XML.

Rendezvous software uses subject-based addressing™ technology to direct messages to the destinations, so program processes can communicate without knowing the details of network addresses or connections. Subject-based addressing conventions define a uniform name space for messages and their destinations.

The locations of component processes become entirely transparent; any application component can run on any network host without modification, recompilation, or reconfiguration. Application programs migrate easily among host computers. You can dynamically add, remove, and modify components of a distributed system without affecting other components.

Subject names consist of one or more elements separated by dot characters (periods). The elements can be used to implement a subject name hierarchy that reflects the structure of information in an application system.

These strings are examples of valid subject names:

`RUN.HOME`

`RUN.for.Elected_office.President`

TIBCO Rendezvous:

- Provides a high performance, scalable platform for e-business infrastructure.
- Enables the creation of robust event-driven applications.
- Harnesses the full capabilities of high performance multi-processor servers.
- Ensures minimal integration-driven traffic as cross system requirements grow.
- Meets the reliability standards of the most demanding applications and 24x7 environments.
- Provides off-the-shelf support for over 100 of the world's leading applications, technologies, and databases.
- Simplifies administration with self-administering protocols.

The BEA WebLogic Adapter for TIBCO

The BEA WebLogic Adapter for TIBCO integrates your TIBCO Rendezvous messages with WebLogic Integration in a fast, easy, and reliable way. You can use the adapter to exchange XML, non-XML, ASCII, and custom data formats between your TIBCO resources and WebLogic Integration to provide a tightly integrated and reliable application infrastructure.

The BEA WebLogic Adapter for TIBCO provides:

- Guaranteed asynchronous, bi-directional message interactions between WebLogic Integration and native TIBCO Rendezvous destinations.
- Data transfer between a business process running within WebLogic Integration and a TIBCO Rendezvous Daemon.
- Service and event adapter integration operations providing end-to-end business process management using XML schemas.
- Support for many formats including:
 - XML
 - Comma Separated Variable (CSV)
 - Excel
 - Message Format Language (MFL)
 - Custom Data Formats (CDF).

The adapter converts non-XML files into XML formats.

Delimited, fixed length, and variable length file formats are supported.

How the BEA WebLogic Adapter for TIBCO Works

The BEA WebLogic Adapter for TIBCO provides transport protocol support so that it can listen for and emit documents from TIBCO queues using TIBCO's daemon. Transaction integrity is maintained at all times. The BEA WebLogic Adapter for TIBCO can accept messages arriving on a named queue and can route these messages to any queue or any other adapter.

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

- Transformation services

XML is quickly becoming the standard for exchanging information between applications and is invaluable in integrating disparate applications. With this in mind, and acknowledging that the world does not yet speak XML exclusively, the BEA WebLogic Adapter for TIBCO provides transformation services.

The adapter uses pre-built customizable parsers to enable the parsing and conversion of non-XML formatted documents and XSLT transformation to modify XML document formats. This ensures that any incoming document can be converted to XML as specified by your event and service schemas.

You can also use the adapter in conjunction with other BEA Adapters to handle the processing of IDoc, SWIFT, FIX, HIPAA, and HL7 message types.

- Document validation rules

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

For example, you can create a name and address validation exit that is called each time such a set of elements appears in the document. The same exit logic might apply to all documents, so that all names and addresses in a complete system are validated in the same manner.

Validation rule specifications are stored in XML files that are freely accessible in the directory structure. Keeping each rule in an external file facilitates the

maintenance of existing rules and provides an easy way to add new ones. You can also create new rules by writing custom Java code.

- **Protocol emitting**

The service adapter supports emitting documents to TIBCO destinations.

- **Document chaining**

You can use any service adapter to enhance the functionality of a document flow. You can chain any number of desired service adapters together.

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

2 Metadata, Schemas, and Repositories

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

This section includes the following topics:

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

Understanding Metadata

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

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

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

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

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

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

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

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

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

Listing 2-1 Data Dictionary Sample

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

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

  <!-- Table 1 -->

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

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

2 *Metadata, Schemas, and Repositories*

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

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

</Segment>

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

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

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

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

After the metadata for your EIS has been described, application views can be created and deployed using the WebLogic Integration Application View Console. For more information on creating application views, see [“Creating an Event Adapter Application View” on page 3-3](#) and [“Creating a Service Adapter Application View” on page 4-1](#).

Schemas and Repositories

You describe all the documents entering and exiting your WebLogic Integration system using W3C XML schemas. These schemas describe each event arriving to and propagating out of an event, and each request sent to and each response received from a service. There is one schema for each event and two for each service (one for the request, one for the response). The schemas are usually stored in files with an `.xsd` extension.

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

BEA WebLogic Adapters all make use of a schema repository to store their schema information and present it to the WebLogic Application View Console. The schema repository is a directory containing:

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

To work with schemas, you must know how to:

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

Naming a Schema Repository

The schema repository has a three-part naming convention:

session_base_directory\adapter\connection_name

- *session_base_directory* is the schema's session base path, which represents a folder under which multiple sessions of schemas may be held.
- *adapter* is the type of adapter (for example, TIBCO or SAP).
- *connection_name* is a name representing a particular instance of the adapter type.

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

`/usr/opt/bea/bse/TIBCO/TIBCODev`

The Repository Manifest

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

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

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

Listing 2-2 Sample Manifest File

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

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

Creating a Repository Manifest

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

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

To create a manifest:

1. Create an XML file with the following structure:

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


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

```
<schemaref name="OrderIn">
  <request root="OrderIn" file="service_OrderIn_request.xsd"/>
  <response root="emitStatus" file="MQEmitStatus.xsd"/>
  <event root="OrderIn" file="event_OrderIn.xsd"/>
</schemaref>
```

Here, the value you assign to:

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

Creating a Schema

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

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

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

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

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

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

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

Listing 2-3 Instance Document for OrderIn Event

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

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

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

Listing 2-4 Schema Matching OrderIn Event Instance Document

```

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

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xsd:element name="OrderIn">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Store_Code"/>
        <xsd:element ref="LineItem" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="LineItem">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Prod_Num"/>
        <xsd:element ref="Quantity"/>
        <xsd:element ref="Price"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Price">
    <xsd:simpleType>
      <xsd:restriction base="xsd:decimal">
        <xsd:enumeration value="1.69"/>
        <xsd:enumeration value="1.79"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="Prod_Num">
    <xsd:simpleType>
      <xsd:restriction base="xsd:short">
        <xsd:enumeration value="1003"/>
        <xsd:enumeration value="1004"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="Quantity">
    <xsd:simpleType>
      <xsd:restriction base="xsd:byte">
        <xsd:enumeration value="10"/>
        <xsd:enumeration value="100"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="Store_Code" type="xsd:hexBinary"/>

```

2 *Metadata, Schemas, and Repositories*

```
</xsd:schema>
```

Storing Directory and Template Files for Transformations

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

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

Samples File

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

3 Creating and Configuring an Event Adapter

This section describes how to create, configure, and test an event adapter application view. An event adapter is the inbound interface from a TIBCO Daemon to a workflow. This section includes the following topics:

- [Creating an Application View Folder](#)
- [Creating an Event Adapter Application View](#)
- [Configuring an Event Adapter Application View](#)
- [Testing Event Adapter Application Views Using 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 a root folder in which you can store all of your application views. You can create additional folders to organize related application views into groups.

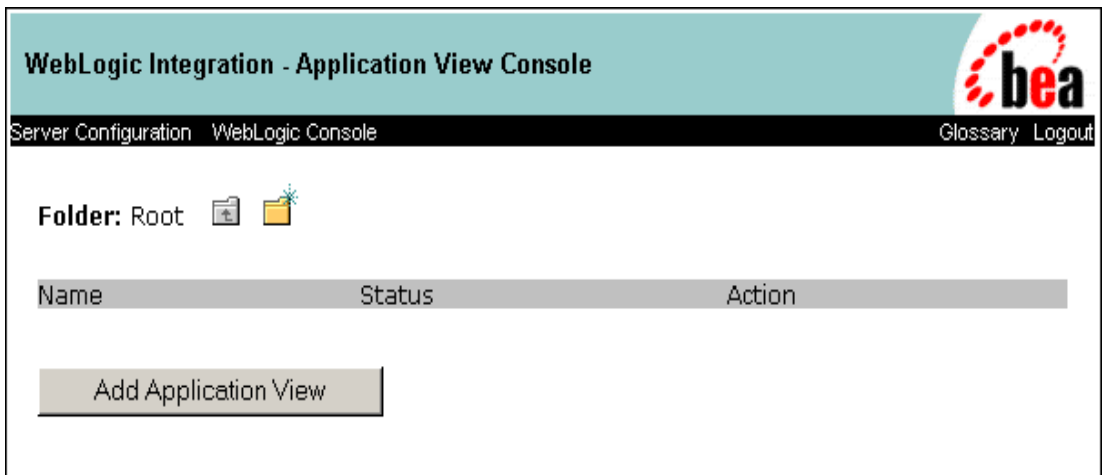
3 Creating and Configuring an Event Adapter

To create an application view folder:

1. Log on to the Application View Console at `//appserver-host:port/wlai`. Here, *appserver-host* is the IP address or host name on which the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.
2. If prompted, enter a user name and password.
Note: If the user name is not `system`, it must be included in the adapter group. For more information on adding the administrative server user name to the adapter group, see the *BEA WebLogic Adapter for TIBCO Installation and Configuration Guide*.
3. Click Login.

The WebLogic Integration Application View Console opens.

Figure 3-1 WebLogic Integration - Application View Console



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

Figure 3-2 Add Folder Window



- b. Enter a name for the folder and click Save.

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

Creating an Event Adapter Application View

To create an event adapter application view:

1. Log on to the Application View Console at `//appserver-host:port/wlai`. Here, *appserver-host* is the IP address or host name on which the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

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

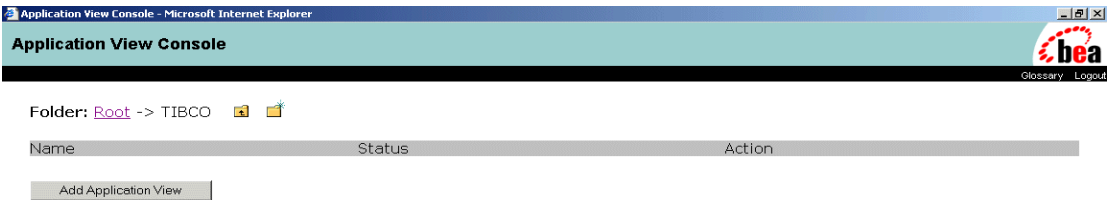
Note: If the user name is not `system`, it must be included in the `adapter` group. For more information on adding the administrative server user name to the `adapter` group, see the *BEA WebLogic Adapter for TIBCO Installation and Configuration Guide*.

3 *Creating and Configuring an Event Adapter*

3. Click Login.

The WebLogic Integration Application View Console opens.

Figure 3-3 Application View Console - Selecting the TIBCO Folder



- a. Select the desired application view folder, for example, TIBCO.
- b. Click Add Application View.

The Define New Application View window opens.

Figure 3-4 Application View Console - Define New Application View

Application View Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Search Favorites History

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

Define New Application View

This page allows you to define a new application view

Folder: TIBCO

Application View Name: *

Description:

Associated Adapter:

OK Cancel

4. In the Define New Application View window, add the following information:
 - a. In the Application View Name field, enter a name.

This name should describe the set of functions performed by this application.

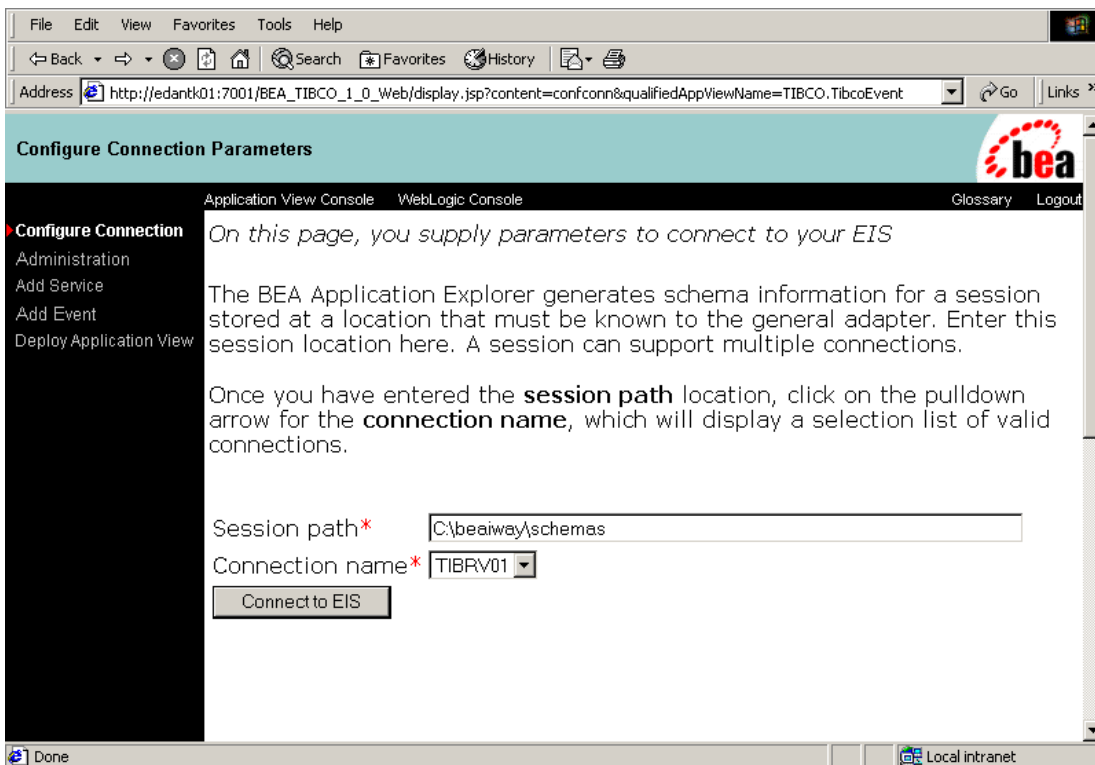
Each application view name must be unique to its adapter. Valid characters include a-z, A-Z, 0-9, and _ (underscore).
 - b. In the Description field, enter any relevant notes. These notes are viewed by users when they use this application view with business process management workflows.
 - c. Select `BEA_TIBCO_1_0` from the Associated Adapter drop-down list.

3 Creating and Configuring an Event Adapter

d. Click OK.

The Configure Connection Parameters window opens.

Figure 3-5 Configure Connection Parameters Window



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

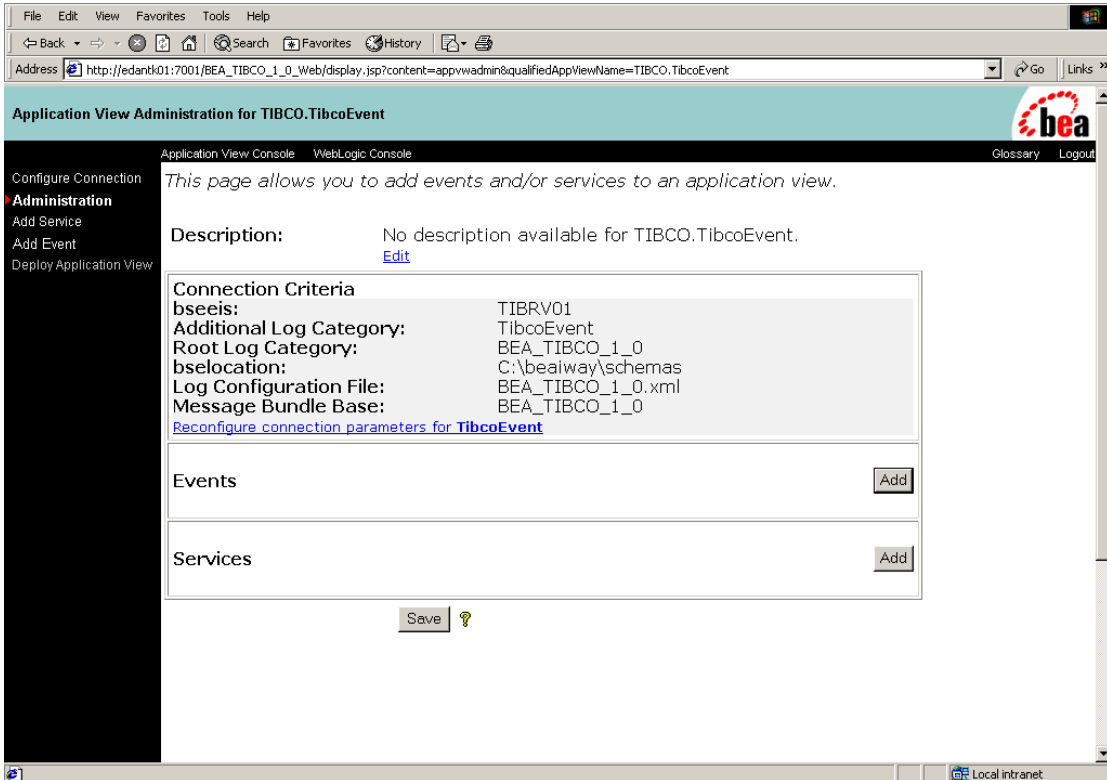
This directory holds your TIBCO schema information and contains the subdirectory `TIBCO/YourConnectionName`.

For example, the session base directory might be `d:\beaiway\schemas`, with the schema repository—containing a repository manifest and schemas—residing in `d:\beaiway\schemas\TIBCO\TIBRV03`. For more information about schema repositories, see [Chapter 6, “Creating Schema Repositories.”](#)

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

The Application View Administration window opens.

Figure 3-6 Application View Administration Window



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

8. Click Save.

You have created 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 8](#), 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 (or document). A service should be added for each event that is used by the application view.

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

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

The Add Event window opens.

Figure 3-7 Add Event Window

Add Event

Application View Console WebLogic Console

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

Unique Event Name: * StatusRpt

Listener: TIBEvent

Daemon	
Network	
service name	
Event Queue	
Send subject	bea.message
Reply subject	bea.subject
Field Name	bea.field
non-XML Preparse	transform(status.xch)
XCH Transform	
XSLT Transform	
encoding	ISO-8859-1
Polling Interval	20

schema: CUST

Add

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

3 Creating and Configuring an Event Adapter

The following table describes these properties.

Table 3-1 Event Properties

Property	Description	Type	Sample Value	Element
non-XML Preparse	Name of the transform type selected for the non-XML to XML transformation phase. Supports Excel, MFL, and .xch transform types. For more information, see Chapter 5, “Transforming Document Formats.”	string	excel	<preparse>
XCH Transform	Name of the .xch transform template file used by the XCH transform type in the XML to XML transformation phase. For more information, see Chapter 5, “Transforming Document Formats.”	string		<in_xmlg>
XSLT Transform	Name of the XSLT stylesheet used by the XSLT transform type in the XML to XML transformation phase. For more information, see Chapter 5, “Transforming Document Formats.”	string		<in_xslt>
Daemon	RV Daemon information to find the TIB/Rendezvous daemon and establish a connection.	string		<Daemon>
Network	Network parameter instructs the Rendezvous daemon to use a particular network for all communications involving this transport. This parameter may be a host name, IP address, or network name. For more information about this parameter, see the TIBCO Rendezvous administration manual. This parameter may be omitted if the daemon is running on the local host.	string		<Network>
Service Name	UDP service on which to listen for TIB/Rendezvous messages. This parameter accepts a service name or a port. The default TIB/Rendezvous port during installation is 7500. If service, network, or daemon are not present, then the listener attempts to connect to a local instance of running TIBCO Rendezvous service. Otherwise, specify the TIBCO Rendezvous instance in the form: <i>host : <port></i>	string		<service>

Table 3-1 Event Properties (Continued)

Property	Description	Type	Sample Value	Element
Event Queue Name	If this parameter is not specified, the listener listens on the Tib/Rendezvous default queue.	string		<queue>
Field Name	Custom field name on which the listener filters.	string		<fieldname>
Send Subject	Each Rendezvous messages bears a subject name. The subject name is used by data consumers to receive all messages labeled with a given name. If the parameter is left blank, the BEA WebLogic Adapter for TIBCO listener listens on subject *. For more information on the send subject parameter, see the topics on Subject Names in the <i>TIB/Rendezvous Concepts</i> manual.	string		<sendsubject>
ReplySubject	Return address to which recipients can send reply messages.	string		<replysubject>
encoding	The character set to use. It defaults to ISO-8869-1, the Latin-1 character set used for most West European languages including English, French, Spanish, German, Dutch, and Swedish.	string	ISO-8859-1	
Polling Interval	The maximum wait interval between checks for new documents. The higher this value, the longer the interval, and the fewer system resources that are used. The side effect of a high value is that the worker thread will not respond to a stop command. If timeout is set to 0, the listener runs once and terminates. Default is 2 seconds.	Duration: xxH:xxM:xxS	1h:2m:3s	

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

4. Click Add.
 5. When the Application View Administration window opens, click Continue.
- The Deploy Application View window opens.

3 Creating and Configuring an Event Adapter

Figure 3-8 Deploy Application View Window

The screenshot shows a web browser window with the address `http://edantk01:7001/BEA_TIBCO_1_0_Web/display.jsp`. The page title is "Deploy Application View TIBCO.TibcoEvent to Server". The left sidebar contains a navigation menu with the following items: "Configure Connection", "Administration", "Add Service", "Add Event", and "Deploy Application View" (which is highlighted). The main content area has a header "On this page you deploy your application view to the application server." and a "Required Event Parameters" section with a text input field for "Event Router URL*" containing `http://edantk01:7001/BEA_TIBCO_1_0_EventRouter/EventRc`. Below this is the "Connection Pool Parameters" section, which includes fields for "Minimum Pool Size*" (1), "Maximum Pool Size*" (10), and "Target Fraction of Maximum Pool Size*" (0.7), along with a checked checkbox for "Allow Pool to Shrink?". The "Log Configuration" section has a dropdown menu set to "Log warnings, errors, and audit messages". The "Configure Security" section contains a link "Restrict Access to TibcoEvent using J2EE Security". At the bottom, there are "Deploy" and "Save" buttons, and a checkbox for "Deploy persistently?" which is checked. The bottom status bar shows "Local intranet".

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

In the WebLogic Server Log, the following entries appear as the event adapter deploys.

Figure 3-9 WebLogic Server Log Window

```

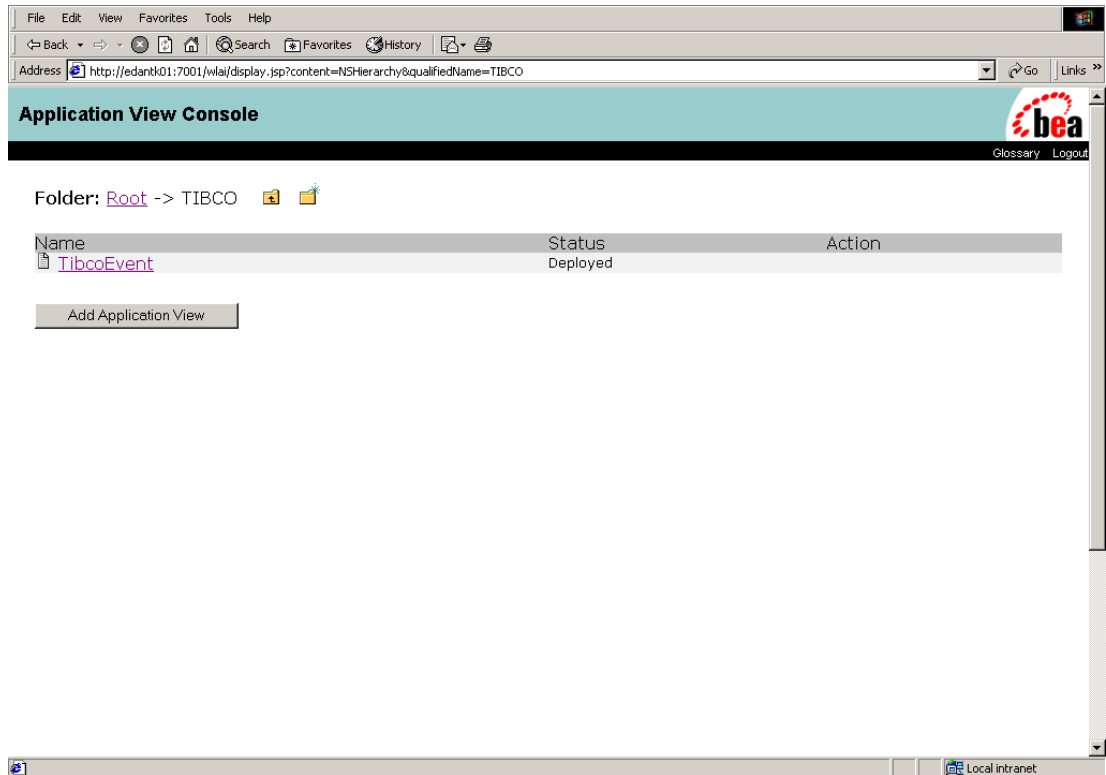
<active>true</active>
</listener>
<system>
  <license>
    <item>external</item>
  </license>
  <define>
    <agent>
      <name value="XDCopyAgent">COPY</name>
      <name value="XDMQEmiAgent">MQEmi</name>
      <name value="XDTransformAgent">transform</name>
    </agent>
    <prepare>
      <name value="XDExcelpreParser">excel</name>
      <name value="XDXMLCpreParser">transform</name>
      <name value="XDMFLpreParser">mfl</name>
    </prepare>
    <xslt>
      <name file="xml2html.xsl">HTML_OUT</name>
    </xslt>
  </define>
  <!-- name file="OrderIn.xsl">OrderIn</name -->
  <settings>
    <debug>true</debug>
    <log>true</log>
    <deepdebug>true</deepdebug>
    <maxlogsize>500</maxlogsize>
    <!-- xmlgroot prompt="root to transform template directory" required="n">transform/xch</xmlgroot>
    <!-- xsltroot prompt="root to XML Style sheet directory" required="n">transform/xslt</xsltroot>
    <xmlgroot>templates</xmlgroot>
    <xsltroot>templates</xsltroot>
  </settings>
</system>
</edaxml>
SetupNewTypes: PushHandler started.
Waiting for startup to complete...
Running iWayPushHandler...
Running XDStart...
DEBUG 01 Jun 2002 22:25:37.583 BEA TIBCO 1_0 - Starting System <version 5.2.1>
DEBUG 01 Jun 2002 22:25:37.583 BEA TIBCO 1_0 - Server code page is Cp1252
DEBUG 01 Jun 2002 22:25:37.583 BEA TIBCO 1_0 - TIBRUEvent: Protocol TIBRU
DEBUG 01 Jun 2002 22:25:37.593 BEA TIBCO 1_0 - TIBRUEvent: retry = [default] 600
DEBUG 01 Jun 2002 22:25:37.593 BEA TIBCO 1_0 - TIBRUEvent: precedence = [dict] 2
DEBUG 01 Jun 2002 22:25:37.593 BEA TIBCO 1_0 - TIBRUEvent: encoding = [dict] ISO-8859-1
DEBUG 01 Jun 2002 22:25:37.593 BEA TIBCO 1_0 - TIBRUEvent: duration = [default] 86400
DEBUG 01 Jun 2002 22:25:37.593 BEA TIBCO 1_0 - TIBRUEvent: agent = [dict] XDCopyAgent
DEBUG 01 Jun 2002 22:25:37.593 BEA TIBCO 1_0 - TIBRUEvent: count = [dict] 1
DEBUG 01 Jun 2002 22:25:37.593 BEA TIBCO 1_0 - TIBRUEvent: fieldname = [dict] bea.field
DEBUG 01 Jun 2002 22:25:37.593 BEA TIBCO 1_0 - TIBRUEvent: sendsubject = [dict] bea.message
DEBUG 01 Jun 2002 22:25:37.593 BEA TIBCO 1_0 - TIBRUEvent: timeout = [default] 2
DEBUG 01 Jun 2002 22:25:37.753 BEA TIBCO 1_0 - Connecting to TibroRsdTransport<null, null, null>
DEBUG 01 Jun 2002 22:25:42.329 BEA TIBCO 1_0 - Queue name : Default Queue
DEBUG 01 Jun 2002 22:25:42.329 BEA TIBCO 1_0 - Queue is the default queue
DEBUG 01 Jun 2002 22:25:42.329 BEA TIBCO 1_0 - queue is a VALID Queue...
DEBUG 01 Jun 2002 22:25:42.399 BEA TIBCO 1_0 - Starting W.TIBRUEvent.1
SetupNewTypes: number of items in eventmap 1
eventtype: <?xml version="1.0"?>
<event name="CustomerProfile"
  rootElementName="customer_profile"
  schemaName="">

```

- To validate that the application view was successfully deployed, invoke the main Application View Console window and select the folder in which you created the application view. You see the name of the new application view with a status of deployed.

3 *Creating and Configuring an Event Adapter*

Figure 3-10 Application View Console - Displaying the Application View Status



You have finished configuring the event adapter application view.

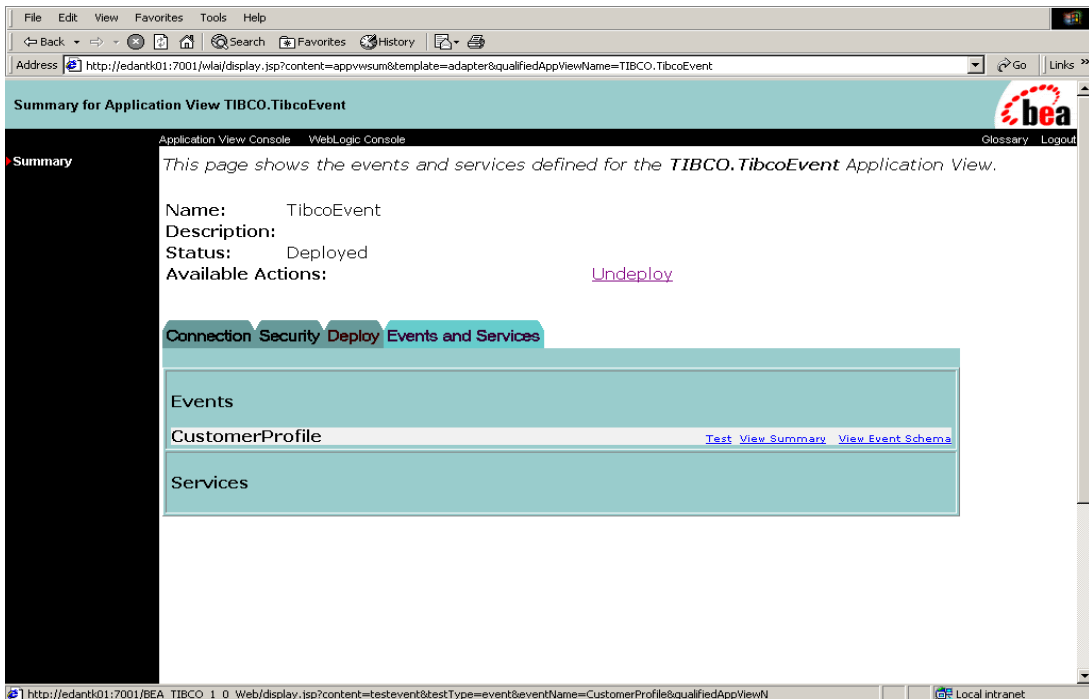
You can confirm that you have configured the event adapter application view correctly, and that it can successfully receive events, using the instructions in “[Testing Event Adapter Application Views Using the Application View Console](#)” on page 15 and “[Testing Event Adapter Application Views Using WebLogic Integration Studio](#)” on page 18.

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 Application View Console as described in “[Creating an Application View Folder](#)” on page 1.
2. Select the folder in which the application view resides and then select the application view. The Summary window opens.

Figure 3-11 Application View - Summary Window

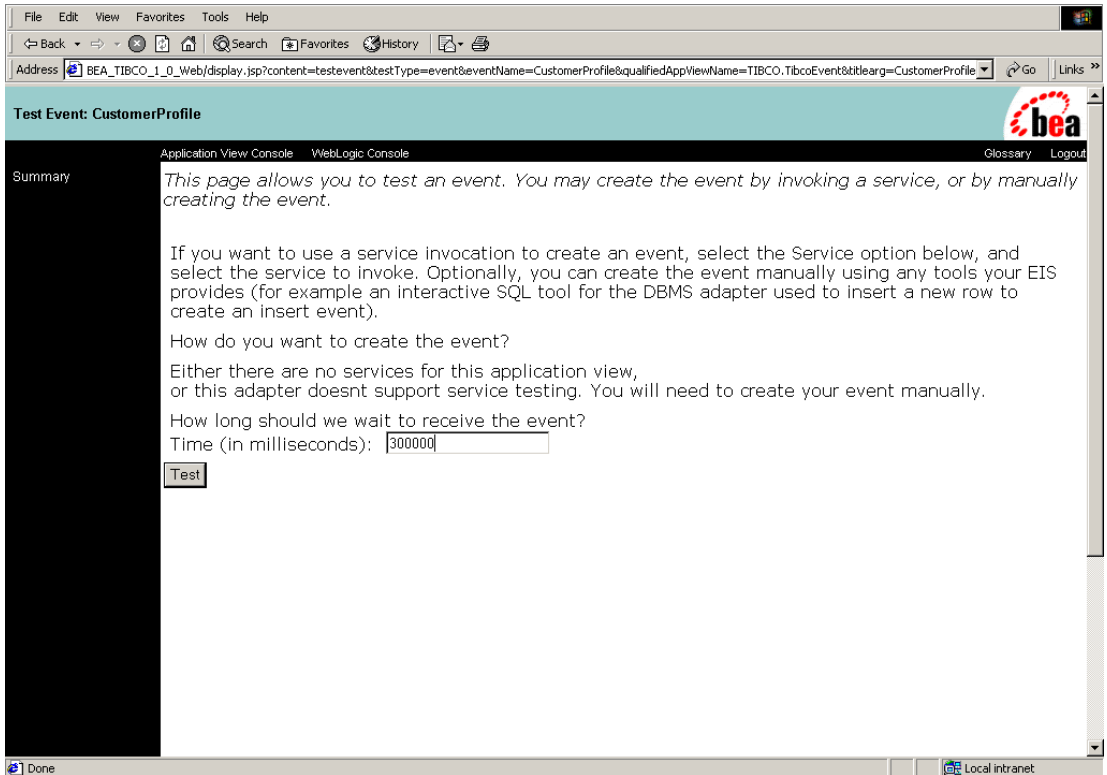


3 Creating and Configuring an Event Adapter

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

The Test Event window opens.

Figure 3-12 Test Event Window



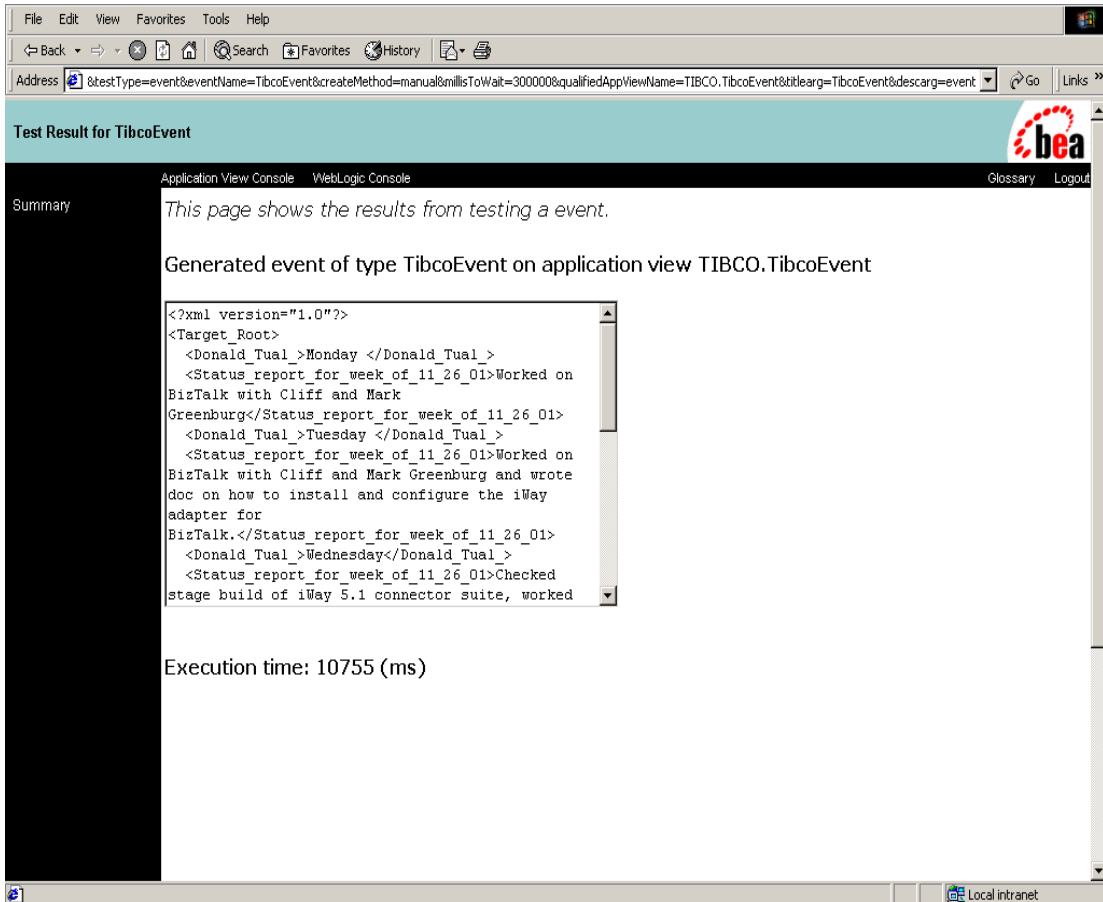
4. Enter 3000 (or a higher value) in the Time field and click Test.

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

5. Click Test.

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

Figure 3-13 Test Result Window



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

Otherwise, you have 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 E-Business Platform→WebLogic Integration 2.1→Studio.

On a UNIX system:

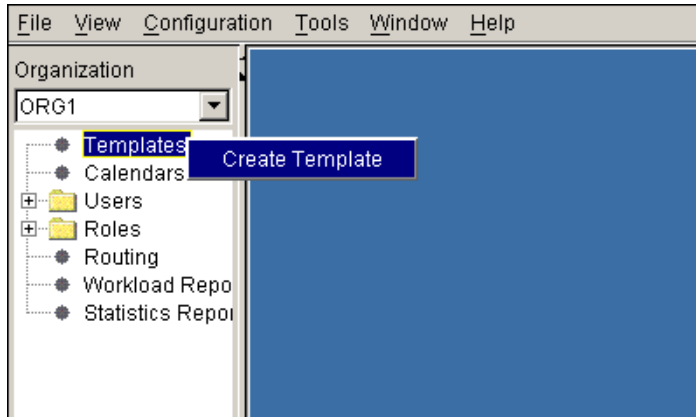
Navigate to the `WLI_HOME/bin` directory and run the `Studio` start-up script by entering the following at the command prompt:

```
sh studio.sh
```

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

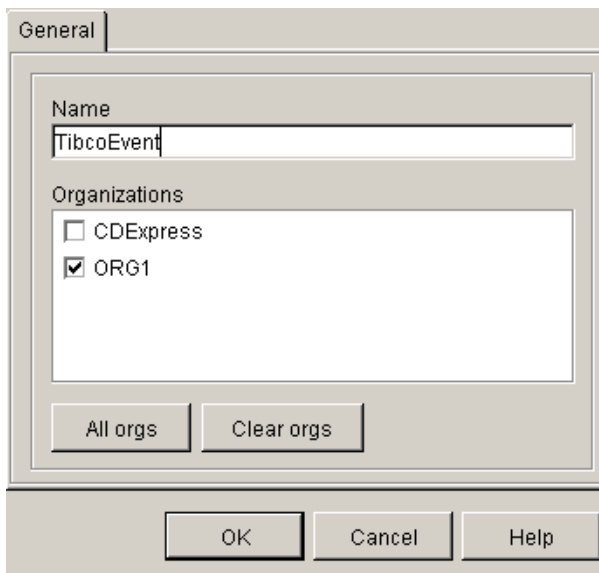
4. Right-click Templates and select Create Template.

Figure 3-14 WebLogic Integration Studio



A Workflow dialog box appears.

Figure 3-15 General Tab of the Workflow Dialog Box



5. Enter a name for your workflow and click OK.

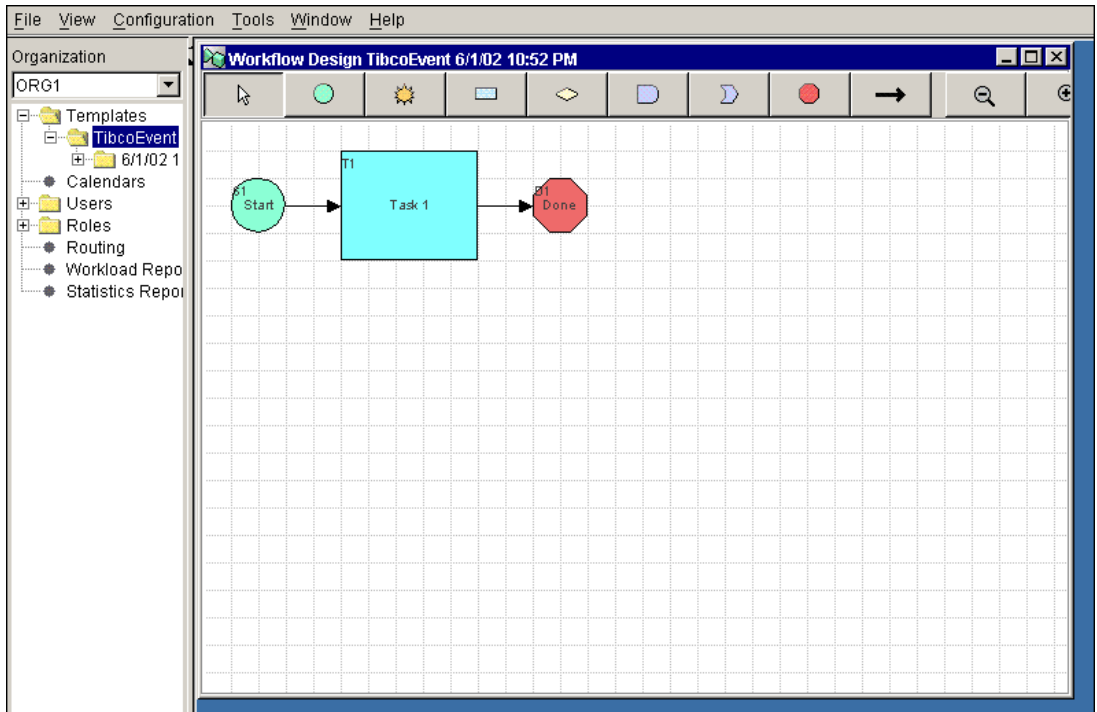
3 *Creating and Configuring an Event Adapter*

In the left pane of WebLogic Integration Studio:

1. Right-click the new template (TibcoEvent) and choose Create Template Definition from the shortcut menu.

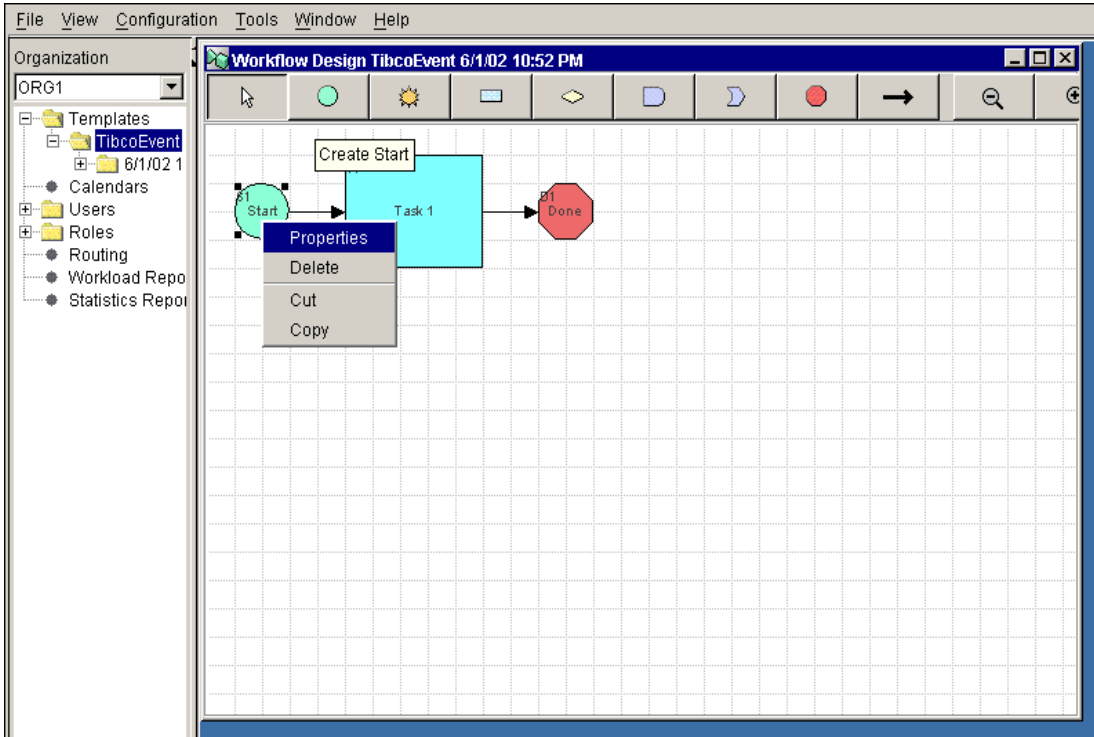
The template opens in WebLogic Integration Studio.

Figure 3-16 Displaying a New Template in WebLogic Integration Studio



2. Right-click the Start node and choose Properties from the shortcut menu.

Figure 3-17 WebLogic Integration Studio - Choosing Node Properties



3 *Creating and Configuring an Event Adapter*

The Start Properties dialog box opens.

Figure 3-18 Start Properties Dialog Box

Description
Start

☐ Timed ☐ Manual ☐ Called ☒ Event All Start

Root
TIBCO
TibcoEvent
CustomerProfile

Name:
CustomerProfile

Description:

Condition:

Event Document Variable:

Refresh Tree View Definition

Start Organization

☐ Use workflow expression

Variables Actions Next Notes

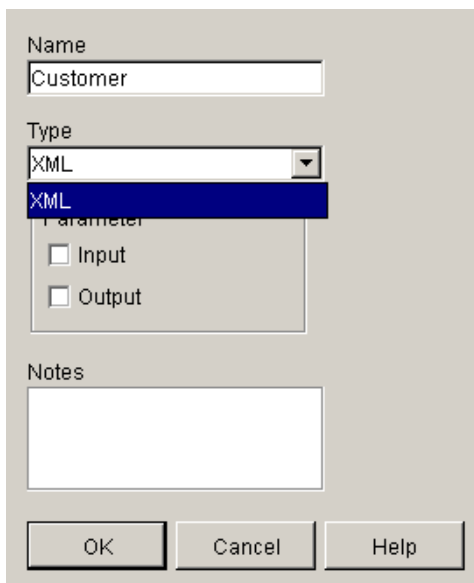
Variable	Expression
----------	------------

Add
Update
Delete

- a. Click the Event option and select AI Start from the drop down list.
- b. In the event explorer, browse the application view folders and select the application view that corresponds to the event adapter.
3. Open the event adapter and select the desired event.
4. Select New from the Event Document Variable drop-down list.

The Variable Properties dialog box opens.

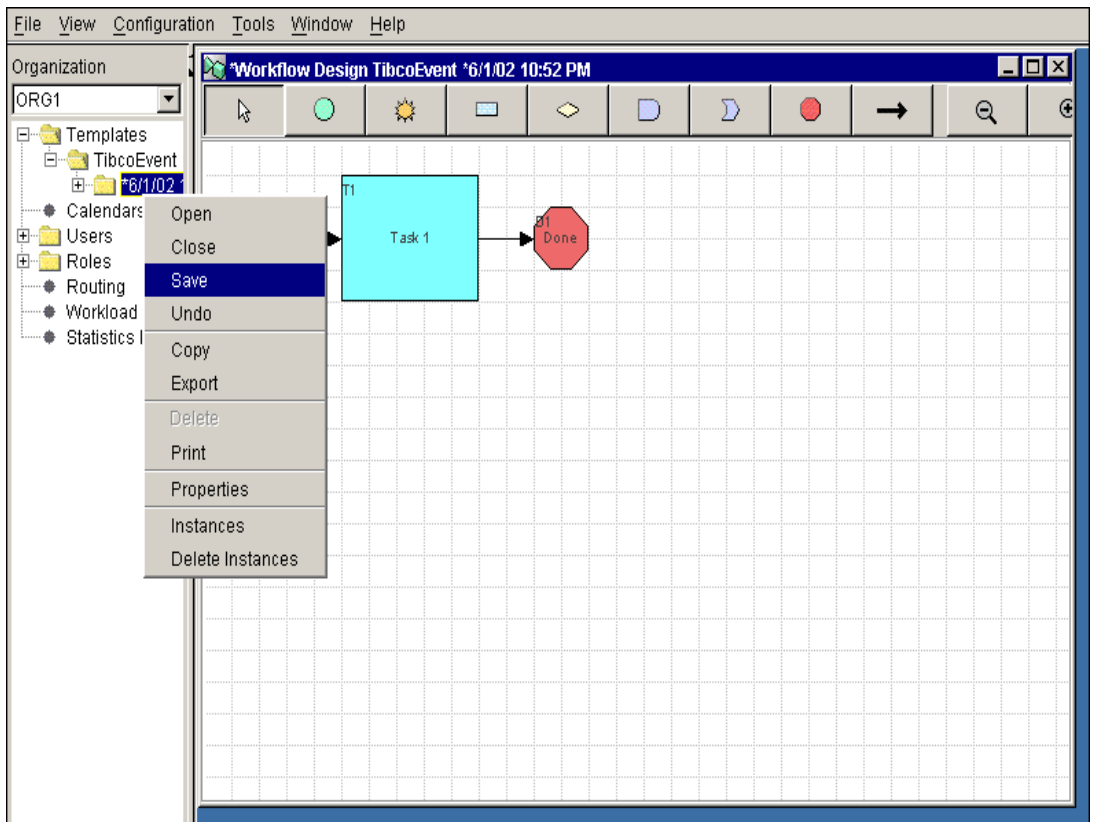
Figure 3-19 Variable Properties Dialog Box

The image shows a 'Variable Properties' dialog box with a light gray background. It contains several fields and controls: a 'Name' text box with 'Customer' entered; a 'Type' drop-down menu with 'XML' selected and a blue highlight bar over the 'XML' option; a 'Parameter' section with two unchecked checkboxes labeled 'Input' and 'Output'; a 'Notes' text area; and three buttons at the bottom: 'OK', 'Cancel', and 'Help'.

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

3 *Creating and Configuring an Event Adapter*

Figure 3-20 WebLogic Integration Studio - Saving a Template



5. Right-click the template in WebLogic Integration Studio's Organization pane and choose Save from the shortcut menu.
6. Right-click the event definition folder and choose Properties from the shortcut menu.

The Template Definition dialog box opens.

Figure 3-21 Template Definition Dialog Box

The screenshot shows a dialog box titled "Template Definition" with two tabs: "General" and "Exception Handlers". The "Exception Handlers" tab is selected. Inside the dialog, there is a "Workflow Label" text field with a blue "A+B" icon to its right. Below this is a checked checkbox labeled "Active". Underneath, there are two date pickers: "Effective" (set to "Jun 1, 2002") and "Expiry" (set to "Dec 31, 2002"). The "Expiry" checkbox is unchecked. Below the date pickers is an unchecked checkbox labeled "Enable auditing". At the bottom of the main content area is a "Notes" text area. At the very bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

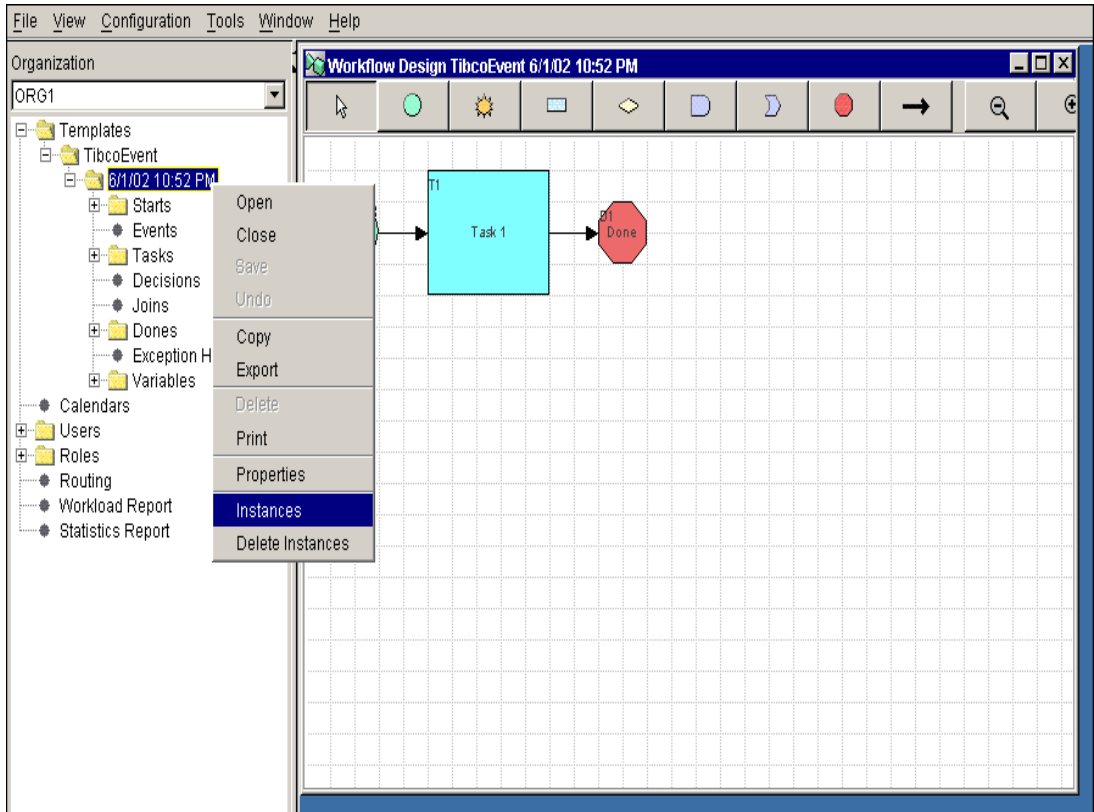
7. Ensure that Active is checked and click OK.

You may now initiate events from your Enterprise Information System (EIS). For the BEA WebLogic Adapter for TIBCO, you can create events through a business application or through a TIBCO Rendezvous test program.

3 *Creating and Configuring an Event Adapter*

8. Return to WebLogic Integration Studio.

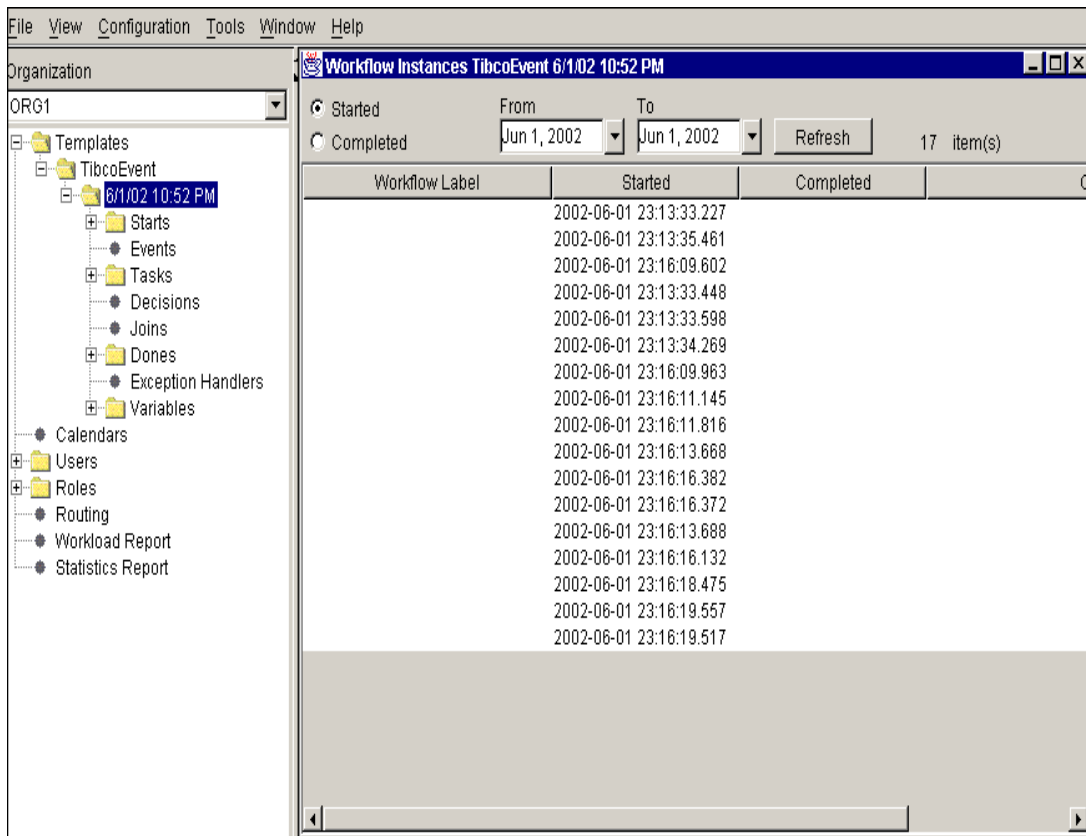
Figure 3-22 WebLogic Integration Studio - Choosing Instances



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

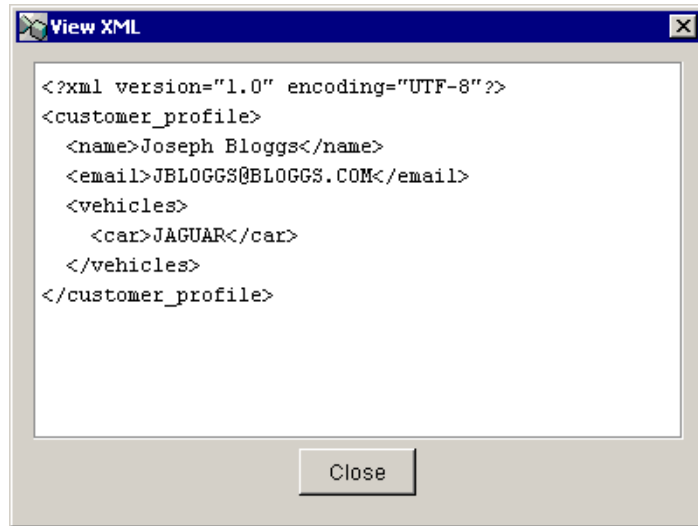
The Workflow Instances for your event definition appear. You can now track the execution of your workflow.

Figure 3-23 WebLogic Integration Studio - Displaying Workflow Instances



- Select Started.
 - Click Refresh.
- A list of workflows appears in the right pane.
- Right-click any instance of the workflow and select Variables.
 - When the Workflow Variables window opens, click View XML to see the entire contents of the workflow message (also known as a document).

Figure 3-24 WebLogic Integration Studio - View XML Window



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

4 Creating and Configuring a Service Adapter

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

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

Creating a Service Adapter Application View

The service adapter for TIBCO is configured for services that send data to TIBCO Rendezvous.

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

4 Creating and Configuring a Service Adapter

Figure 4-1 Define New Application View Window

The screenshot shows a web browser window titled "Application View Console - Microsoft Internet Explorer". The address bar shows the URL "http://edantk01:7001/wla/display.jsp?content=defappvw&namespace=". 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 elements:

- A link labeled "TIBCO" in purple text.
- A label "Application View Name: *" followed by a text input field containing "TibcoService".
- A label "Description:" followed by a large text area.
- A label "Associated Adapter:" followed by a dropdown menu showing "BEA_TIBCO_1_0".
- An "Cancel" button.

The browser's status bar at the bottom indicates "Local intranet".

2. In the Define New Application View window, add the following information:
 - a. In the Application View Name field, enter a name.

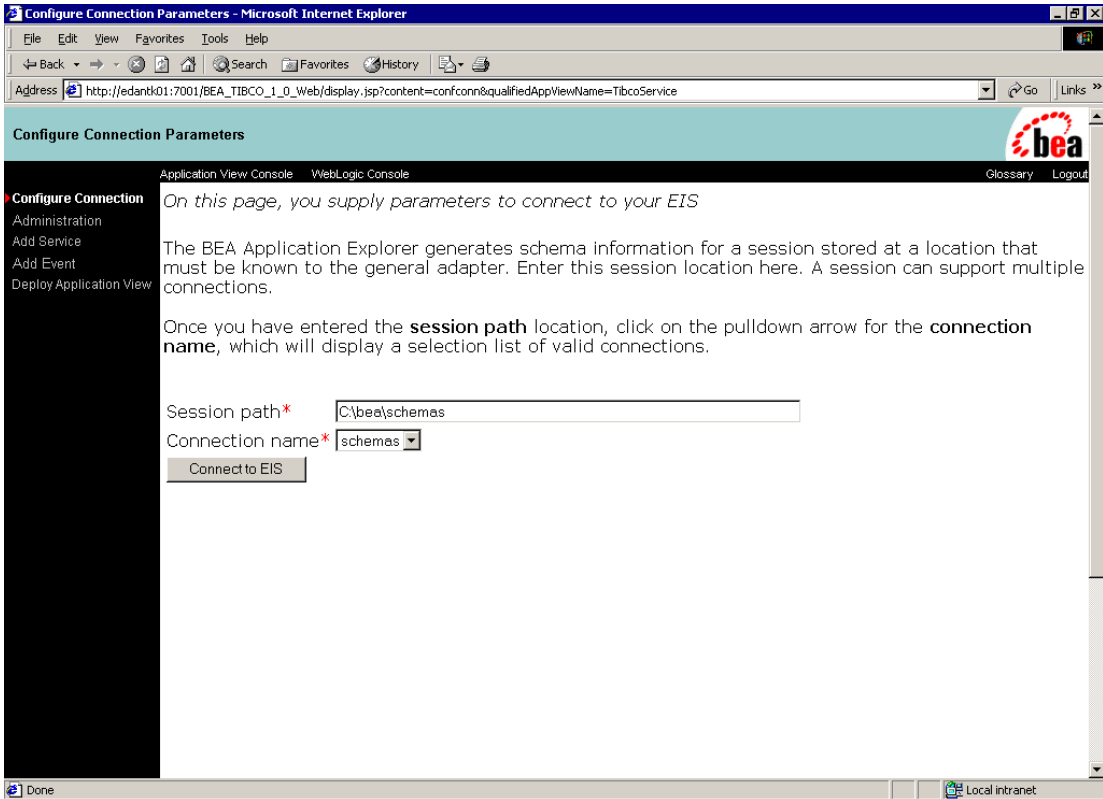
This name should describe the set of functions performed by this application.

Each application view name must be unique to its adapter. Valid characters include a-z, A-Z, 0-9, and _ (underscore).
 - b. In the Description field, enter any relevant notes. These notes are viewed by users when they use this application view with business process management workflows.
3. Select BEA_TIBCO_1_0 from the Associated Adapter list.

4. Click OK.

The Configure Connection Parameters window opens.

Figure 4-2 Configure Connection Parameters Window



5. Enter the name of the BEA WebLogic Adapter for TIBCO session base directory in the Session path field. This directory holds your TIBCO schema information and contains the subdirectory *TIBCO/YourConnectionName*.

For example, the session base directory might be

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

d:\bea\bse\sessions\default\TIBCO\TIBRV01. For more information about schema repositories, see [Chapter 6, “Creating Schema Repositories.”](#)

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

Note that you can access the Configure Connection Parameters window 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 window by first undeploying the application view.

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

Configuring a Service Adapter Application View

To configure a service adapter application view:

1. Log on to the Application View Console at

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

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

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

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

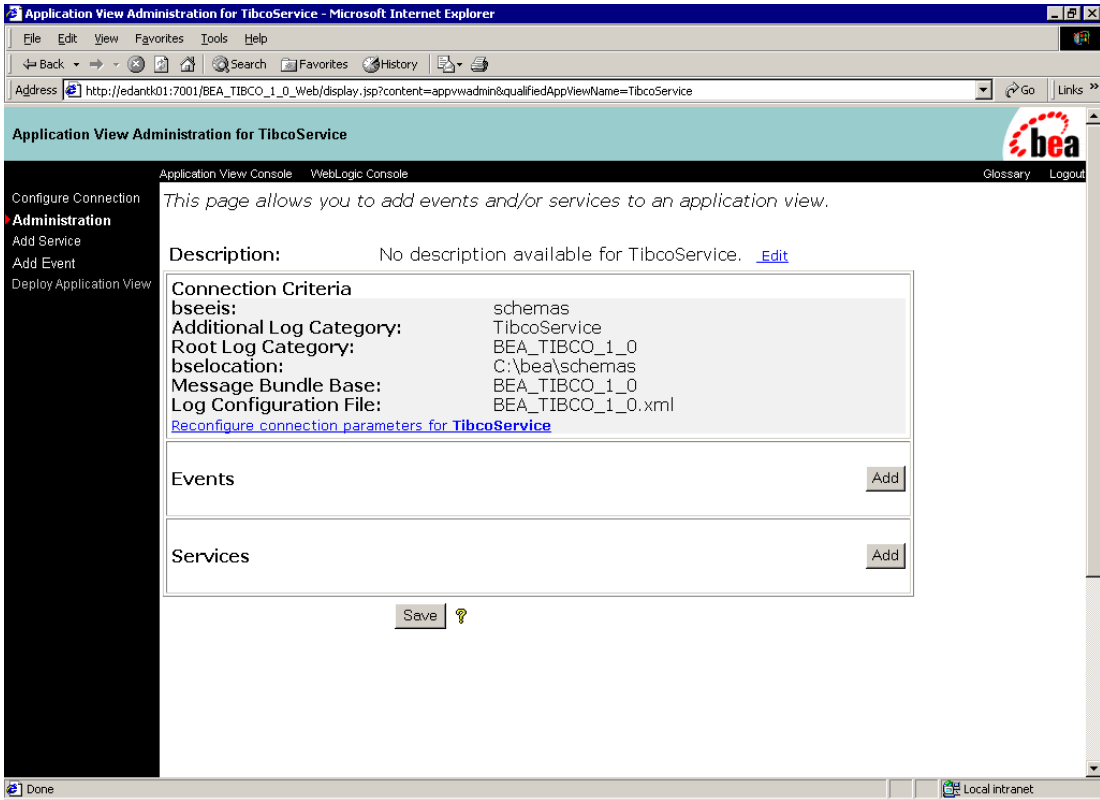
3. Click Login.

The WebLogic Integration Application View Console opens.

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

The Administration window opens.

Figure 4-3 Application View Administration Window



5. Click Add in the Services pane.
The Add Service window opens.

Figure 4-4 Add Service Window

Add Service

Application View Console WebLogic Console

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

Unique Service Name:

TIBService

Transform name	mflout
type	flat
Transform engine	mfl
Broker Host	EDANTK01
Broker Port	7500
Send Subject	bea.send.message
Reply Subject	
Field Name	bea.send.field

settings

Trace on/off	<input type="checkbox"/>
Logging on/off	<input type="checkbox"/>
root to transform template directory	transform/txch
root to XML Style sheet directory	transform/xslt

schema:

- Update the required properties of TIBCO Rendezvous as described in the following table.

These settings correspond to the TIBCO communication settings that the service adapter uses to communicate with TIBCO Rendezvous to publish messages to the queues, and with the transform options described in [Chapter 5, “Transforming Document Formats.”](#)

Table 4-1 Service Properties

Property	Description	Type	Sample Value
Transform Name	Name of the transform template file used by the transform engine in either XSLT, XCH, or MFL transformations. For MFL transformations, do not enter an .mfl extension; these files are not stored with an extension. For more information, see Chapter 5, “Transforming Document Formats.”	string	
Type	File format for the output (flat or xml).	string	
Transform Engine	Name of the transform engine to perform the transformation. This field can be omitted if no transformations are performed.	string	
Broker Host	Host where RVD daemon is listening.	string	Localhost
Broker Port	Port on which RVD daemon is listening	String	7500
Send Subject	Each Rendezvous message bears a subject name. The subject name is used by data consumers to receive all messages labeled with a given name. If the parameter is left blank, the BEA WebLogic Adapter for TIBCO emitter defaults the send subject for the TIBCO Rendezvous message to IWAYDEFAULTSUBJECT.	string	
Reply Subject	Return address, to which recipients can send reply messages. If this parameter is omitted, no default value is supplied.	string	
Field Name	A field name is a character string. Each field can have a maximum of one name. Several fields can have the same name. If the fieldname parameter is not specified, the TIBCO Rendezvous message with the payload associated with it is emitted with the default field name of IWAYDEFAULTOBJECT.	duration	

4 Creating and Configuring a Service Adapter

The schema drop-down list corresponds to the manifest that describes all schemas associated with the view.

- b. Select the schema you wish to work with.
- c. If desired, enable trace and logging.
- d. Click Add.
- e. When the Administration window opens, click Continue.

The Deploy Application View window opens.

Figure 4-5 Deploy Application View Window

Deploy Application View TibcoService to Server

Application View Console WebLogic Console Glossary Logout

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

Required Service Parameters

Enable asynchronous service invocation? ☒

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

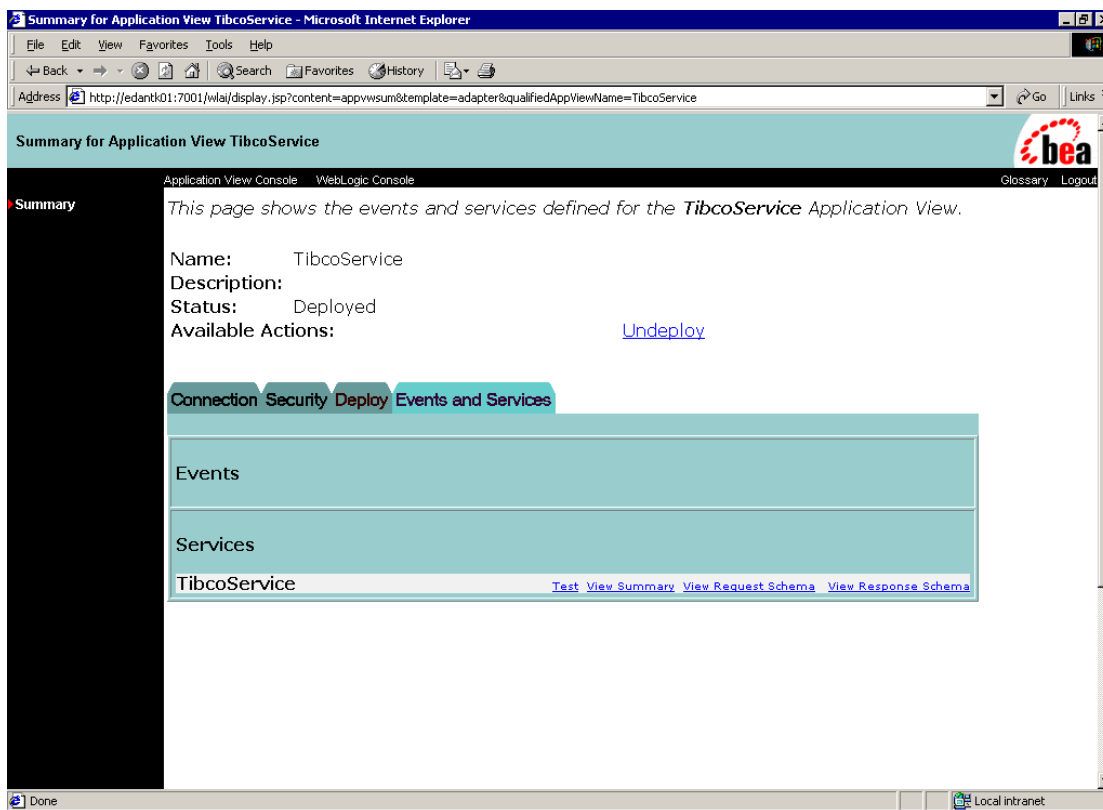
[Restrict Access to TibcoService using J2EE Security](#)

☒ Deploy persistently?

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

The Summary for Application View TibcoService window opens on successful deployment.

Figure 4-6 Summary for Application View Window



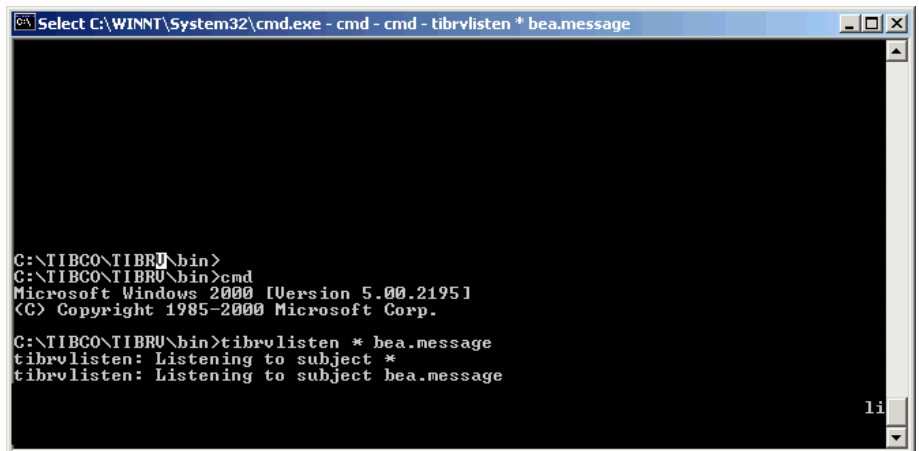
You have finished configuring the service adapter application view. You can confirm that you configured it correctly by following the instructions in [“Testing the BEA Service Adapter for TIBCO”](#) on page 10 and [“Testing Service Adapter Application Views Using WebLogic Integration Studio”](#) on page 15.

Testing the BEA Service Adapter for TIBCO

The service adapter publishes a document to TIBCO and returns an emit status. You can validate the document’s arrival on the queue using the TIBCO-supplied TIBRVLISTEN program.

1. Open a command window to the TIBCO Rendezvous bin directory.

Figure 4-7 TIBRVLISTEN Command Window



```
Select C:\WINNT\System32\cmd.exe - cmd - cmd - tibrvlisten * bea.message

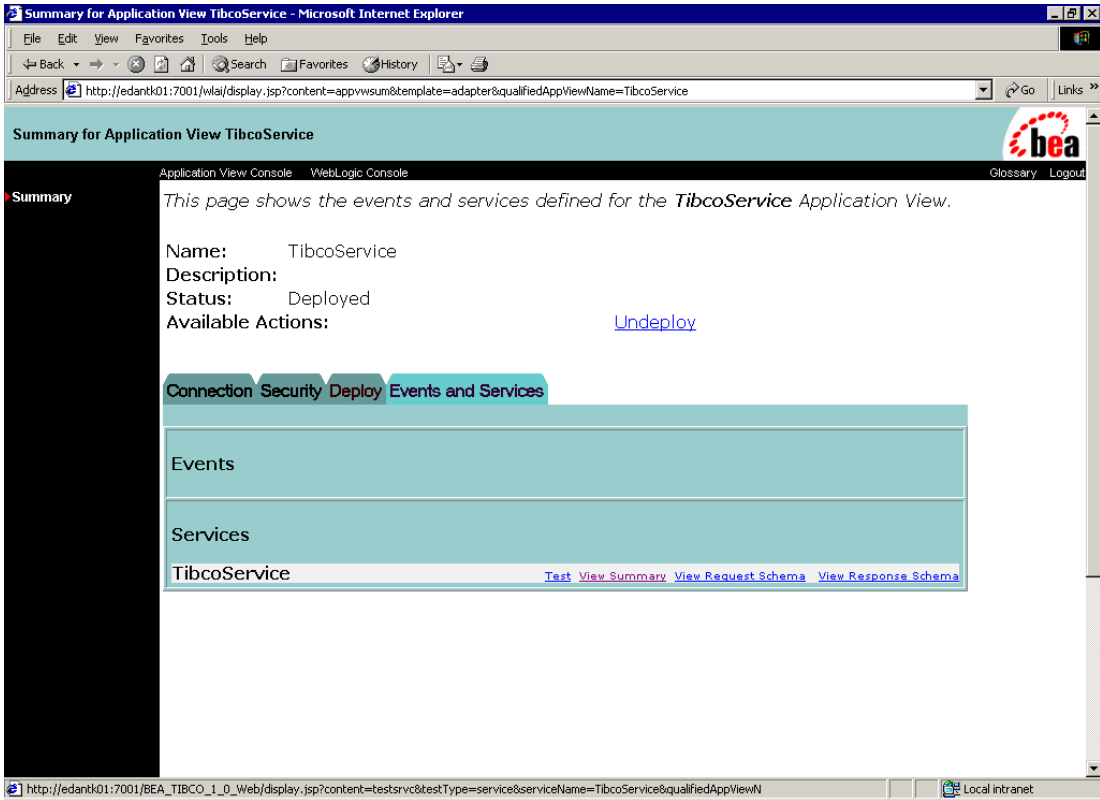
C:\TIBCO\TIBRV\bin>
C:\TIBCO\TIBRV\bin>cmd
Microsoft Windows [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\TIBCO\TIBRV\bin>tibrvlisten * bea.message
tibrvlisten: Listening to subject *
tibrvlisten: Listening to subject bea.message
```

2. Start the TIBCO Rendezvous listen program to listen on the Send Subject of bea.message.
3. Issue the command TIBRVLISTEN * bea.message.

The Summary for Application View window opens.

Figure 4-8 Summary for Application View Window

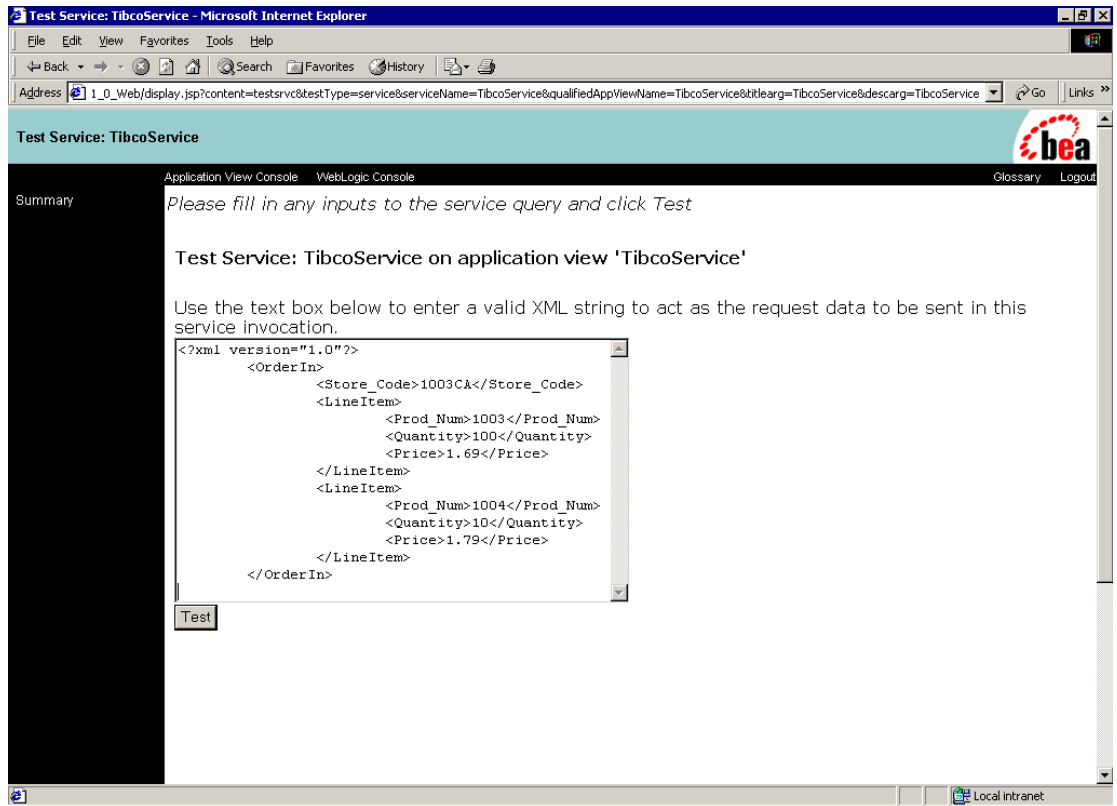


4. Click Test.

The Test Service window opens.

4 Creating and Configuring a Service Adapter

Figure 4-9 Test Service Window - Entering Sample Document



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

For example, the `OrderIn` request schema contains an instance document such as the following code listing.

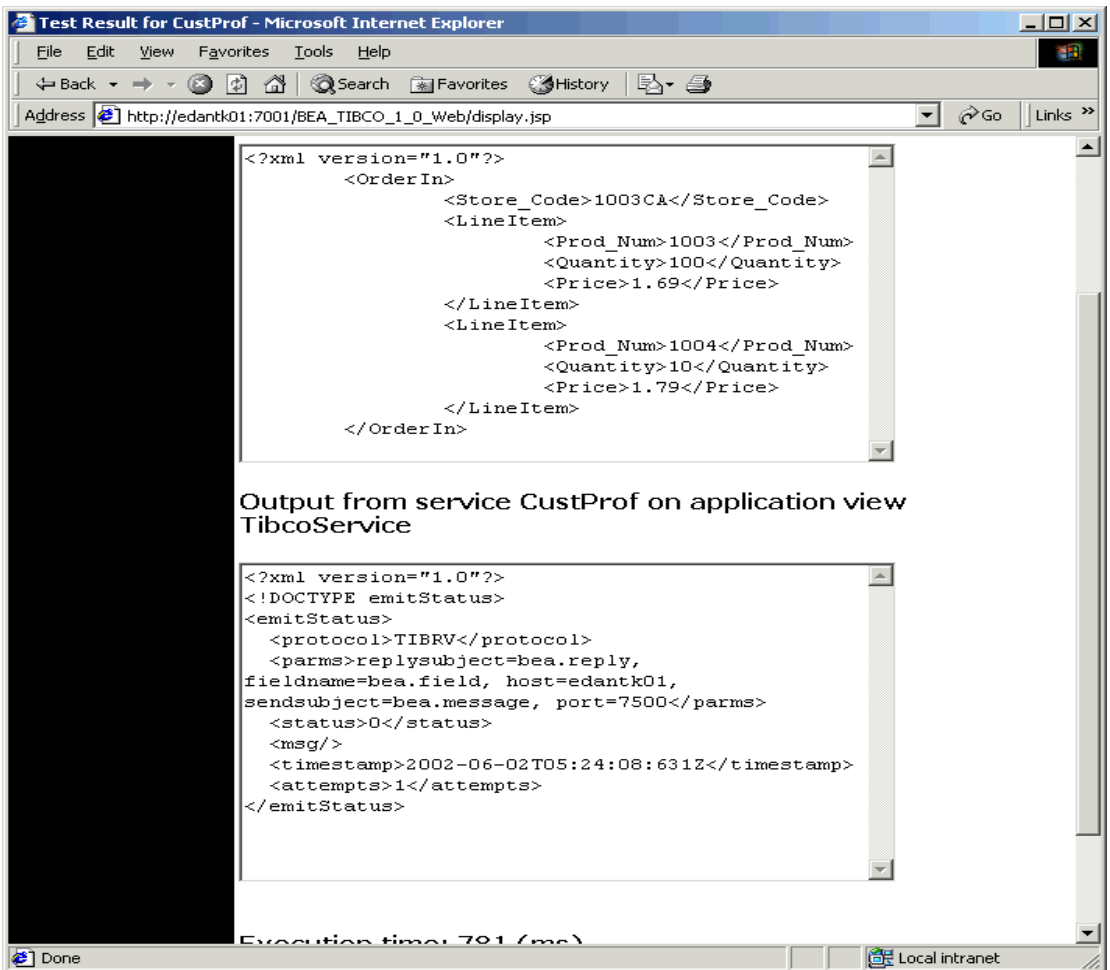
Listing 4-1 Instance Document for OrderIn Request Schema

```
<?xml version="1.0"?>
<OrderIn>
  <Store_Code>1003CA</Store_Code>
  <LineItem>
    <Prod_Num>1003</Prod_Num>
    <Quantity>100</Quantity>
    <Price>1.69</Price>
  </LineItem>
  <LineItem>
    <Prod_Num>1004</Prod_Num>
    <Quantity>10</Quantity>
    <Price>1.79</Price>
  </LineItem>
</OrderIn>
```

6. Enter this document into the Service Test window by either typing it, or copying and pasting it, into the window.
7. Click Test to send the request through the service adapter to TIBCO Rendezvous.

The response document indicates the success of emitting to TIBCO. See the sample response in the following test result window.

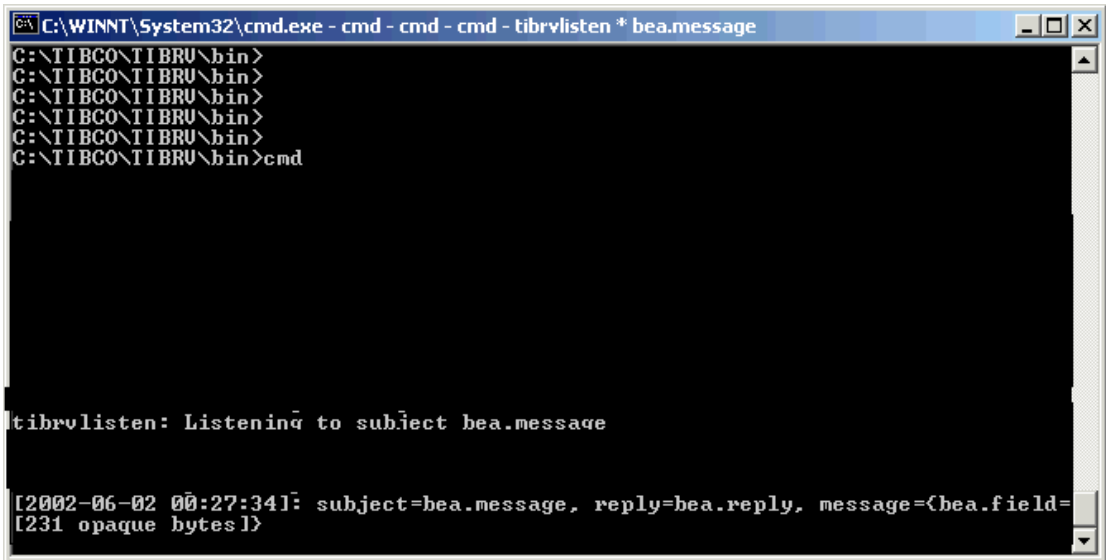
Figure 4-10 Test Result Window



After transformation and formatting, the document is published to TIBCO Rendezvous.

8. Return to the TIBRVLISTEN command window to verify that the TIBCO message has arrived.

Figure 4-11 TIBRVLISTEN Command Window - Verifying the Message



```
C:\WINNT\System32\cmd.exe - cmd - cmd - cmd - tibrvlisten * bea.message
C:\TIBCO\TIBRU\bin>
C:\TIBCO\TIBRU\bin>
C:\TIBCO\TIBRU\bin>
C:\TIBCO\TIBRU\bin>
C:\TIBCO\TIBRU\bin>
C:\TIBCO\TIBRU\bin>cmd

tibrvlisten: Listening to subject bea.message

[2002-06-02 00:27:34]: subject=bea.message, reply=bea.reply, message={bea.field=
1231 opaque bytes}
```

Testing Service Adapter Application Views Using WebLogic Integration Studio

To confirm that a deployed service adapter application view is correctly configured and can receive events, test the adapter application using WebLogic Integration Studio.

1. Start the WebLogic Integration Studio:

On a Windows system, choose Start→Programs→BEA WebLogic E-Business Platform→WebLogic Integration 2.1→Studio.

On a UNIX system, go to the `WLI_HOME/bin` directory and run the `Studio` start-up script by entering the following at the command prompt:

```
sh studio.sh
```

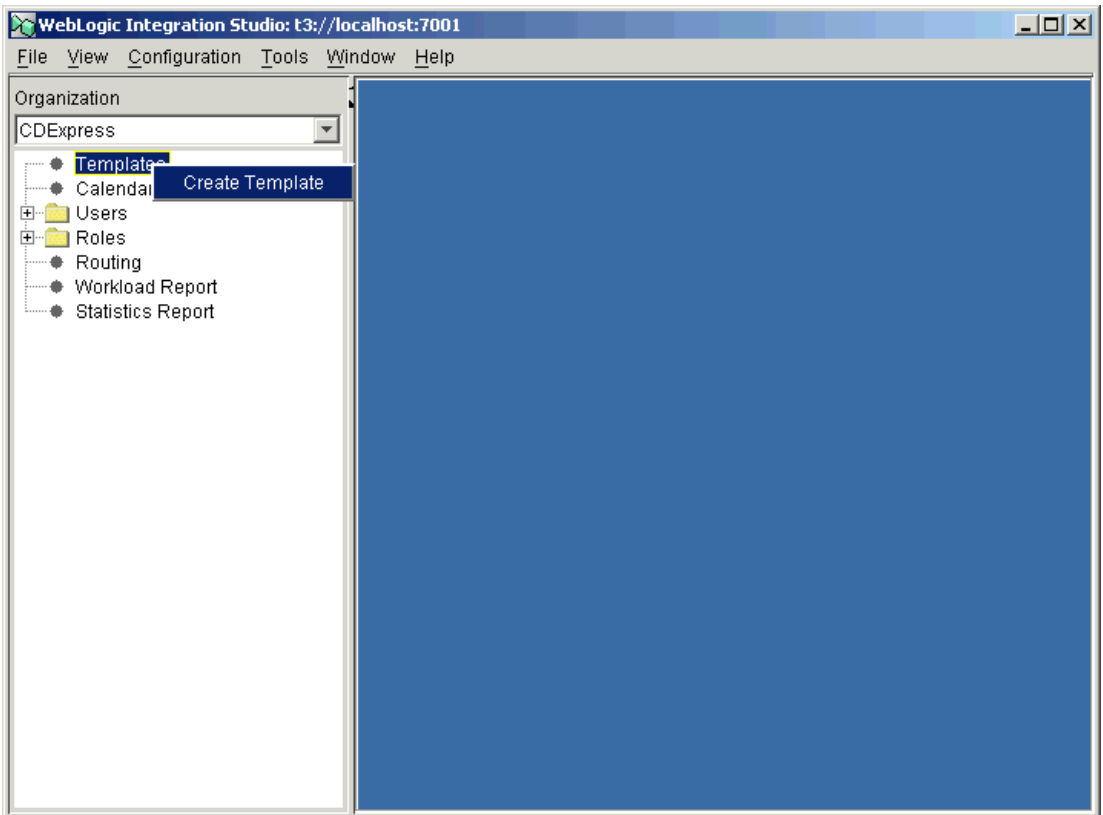
2. Log on to the WebLogic Integration Studio.

4 *Creating and Configuring a Service Adapter*

3. In the Organization pane, select an organization to create a new workflow template.

The WebLogic Integration Studio window opens.

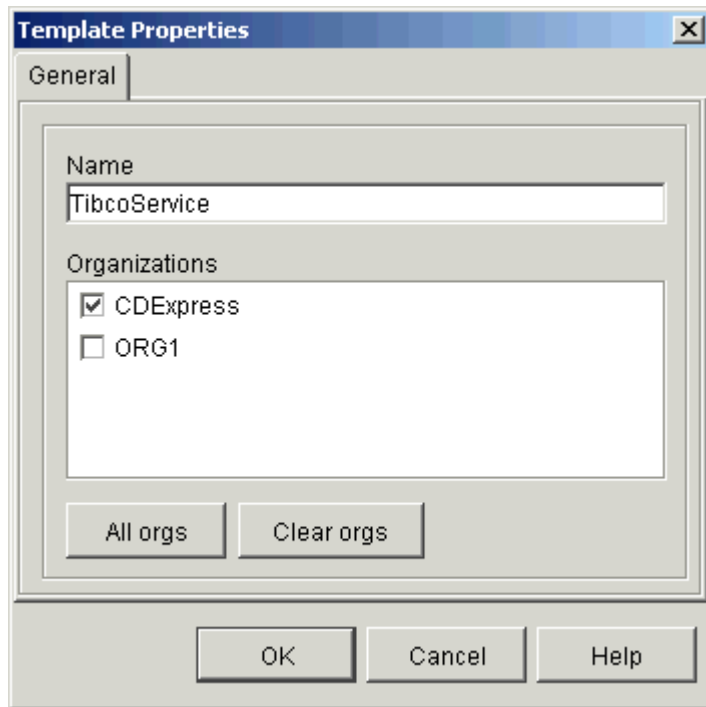
Figure 4-12 WebLogic Integration Studio



4. Right-click Templates and choose Create Template.

The Template Properties dialog box appears.

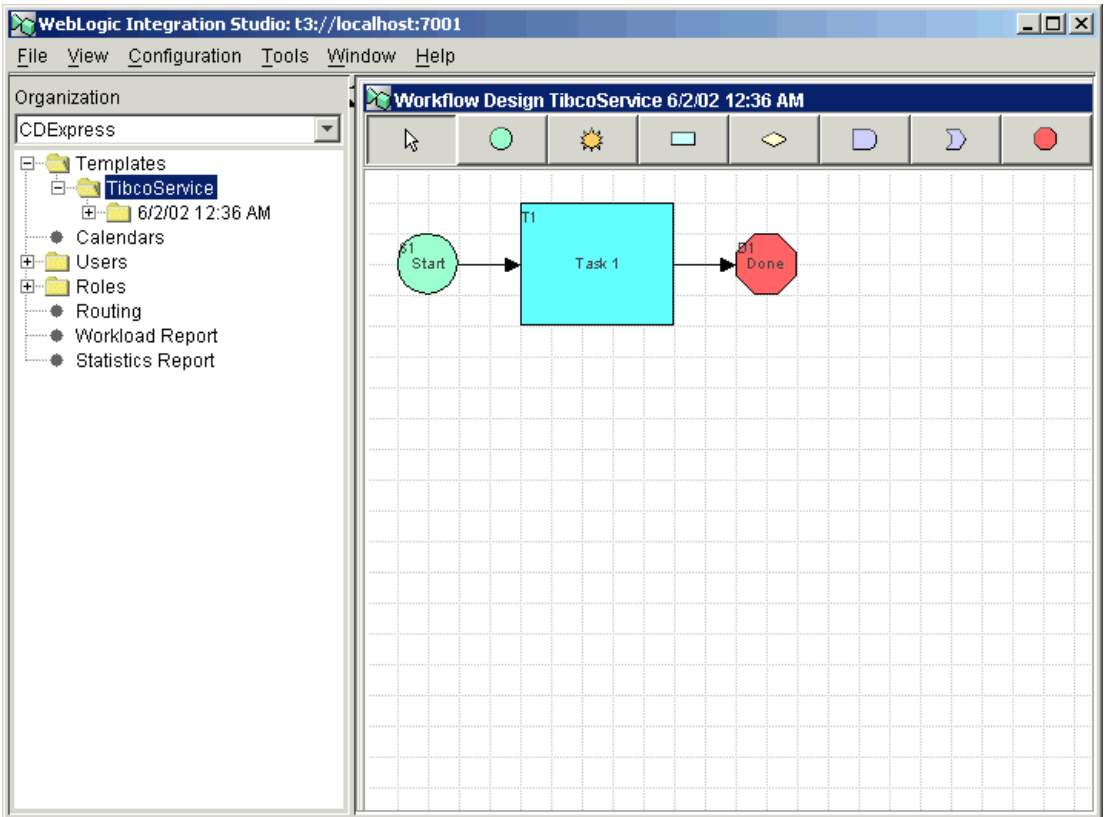
Figure 4-13 Template Properties Dialog Box



- a. In the Template Properties dialog box, enter a name for your workflow template.
 - b. Click OK.
5. Right-click the new template and choose Create Template Definition.

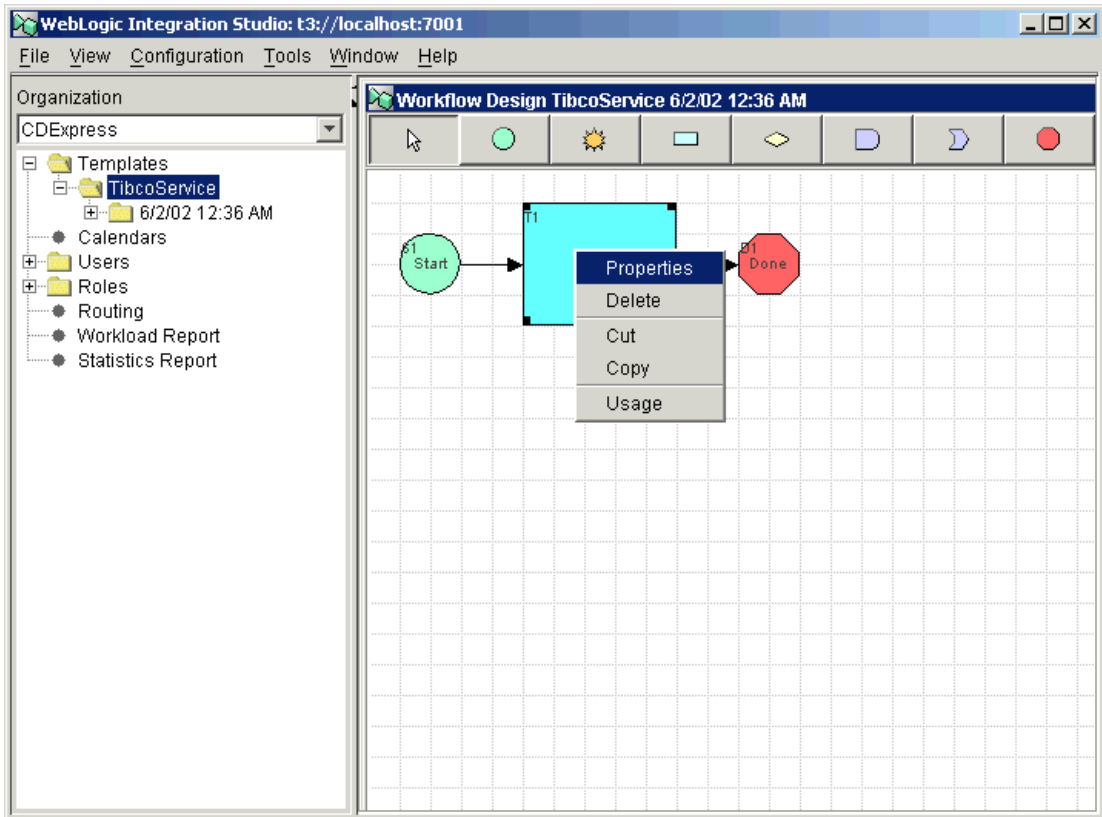
4 *Creating and Configuring a Service Adapter*

Figure 4-14 WebLogic Integration Studio - Displaying a New Template



The template appears in WebLogic Integration Studio.

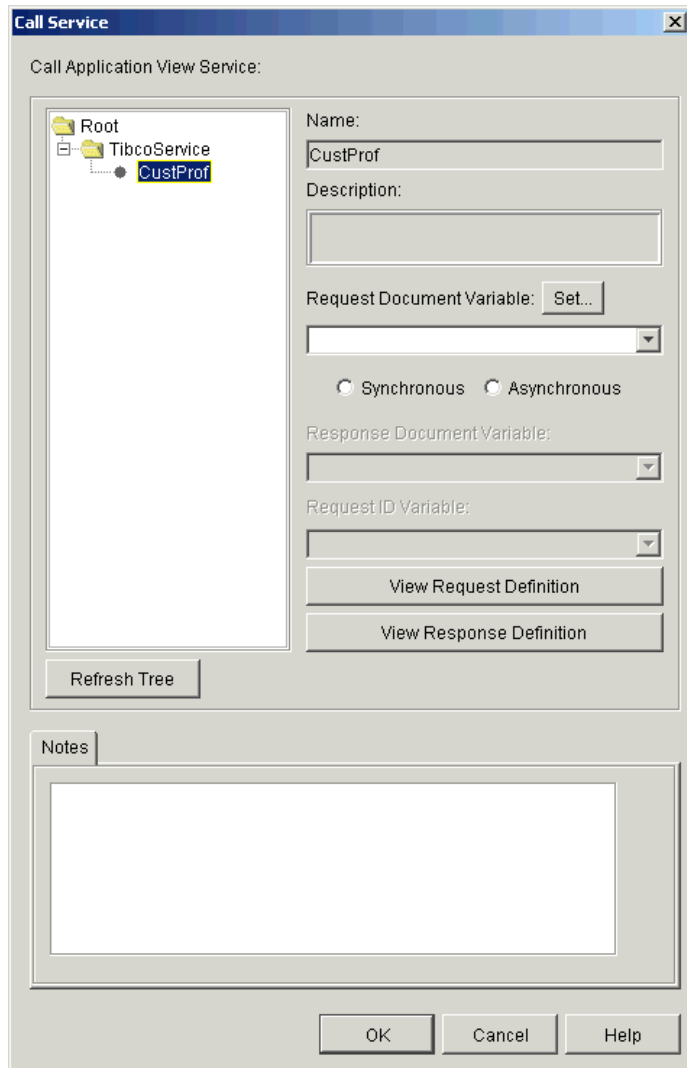
Figure 4-15 WebLogic Integration Studio - Choosing Node Properties



6. Right-click the Task node and choose Properties.
7. After the Start Properties dialog box appears, select Add, AI Actions, and Call Application View Service.

The Call Service dialog box appears.

Figure 4-16 Call Service Dialog Box

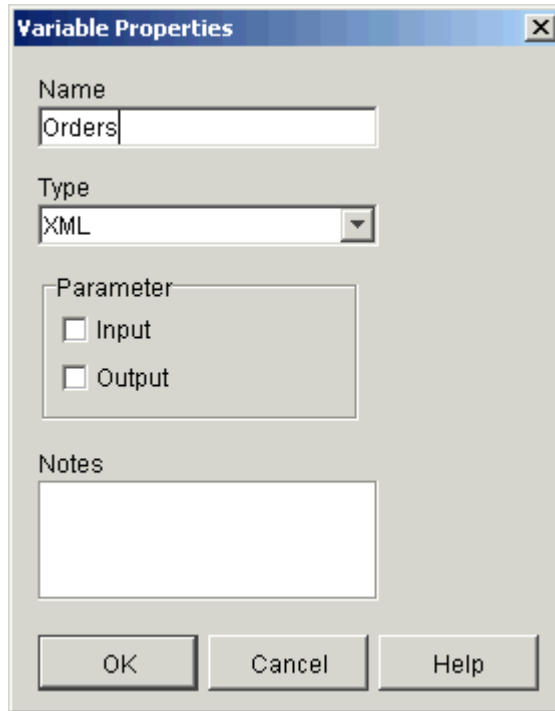


- In the service explorer, browse the Application View folders and select the application view that corresponds to the service adapter.
- Open the service adapter and select the desired service.

- c. Select <new> from the Request Document Variable drop-down list.

The Variable Properties dialog box opens.

Figure 4-17 Variable Properties Dialog Box

The image shows a 'Variable Properties' dialog box with a title bar containing a close button. The dialog has several sections: 'Name' with a text field containing 'Orders'; 'Type' with a dropdown menu showing 'XML'; 'Parameter' with two checkboxes, 'Input' and 'Output', both of which are unchecked; and 'Notes' with a large empty text area. At the bottom, there are three buttons: 'OK', 'Cancel', and 'Help'.

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

You are returned to the Call Service dialog box.

- a. Select Synchronous.
- b. Select New from the Response Document Variable drop-down list.
- c. Enter a name for the new variable.

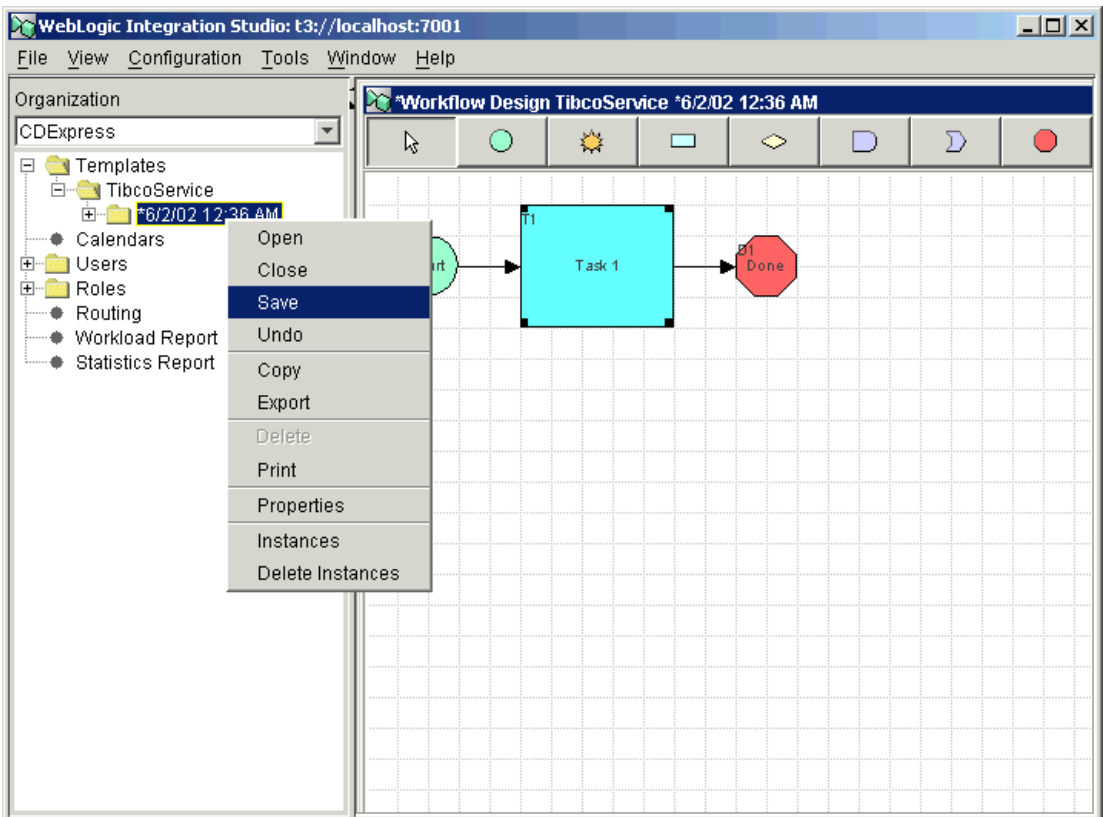
4 *Creating and Configuring a Service Adapter*

- d. Select the variable type XML.

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

- e. Click Set to set the request document. This enables you to choose an XML document containing the service request.
- f. Click OK to return to the template.

Figure 4-18 WebLogic Integration Studio - Saving a Template

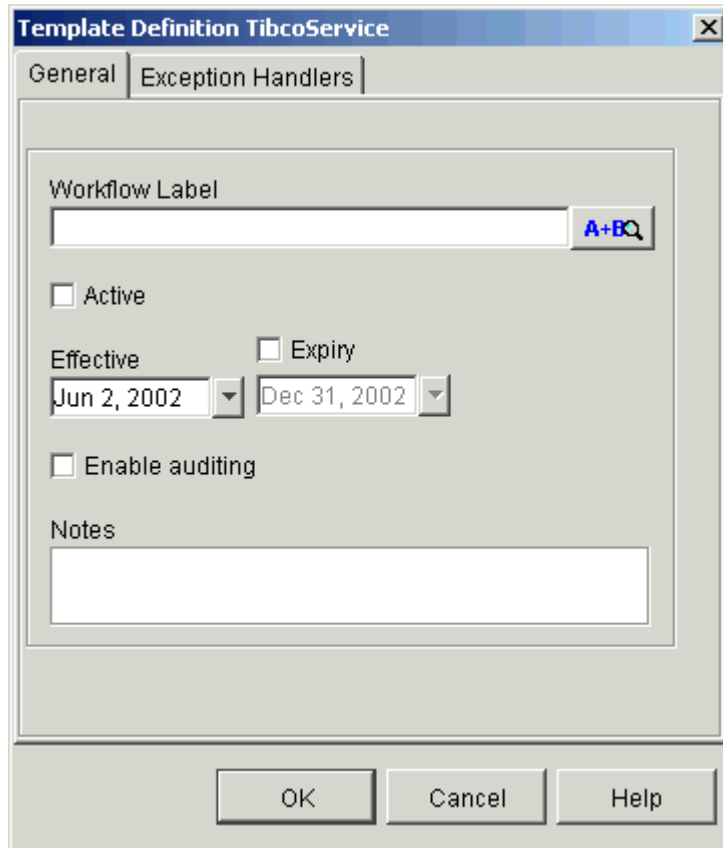


9. Right-click the event definition in WebLogic Integration Studio's Organization pane and select Save.

10. Right-click the event definition folder and choose Properties.

The Template Definition dialog box appears.

Figure 4-19 Template Definition Dialog Box



a. Click Active.

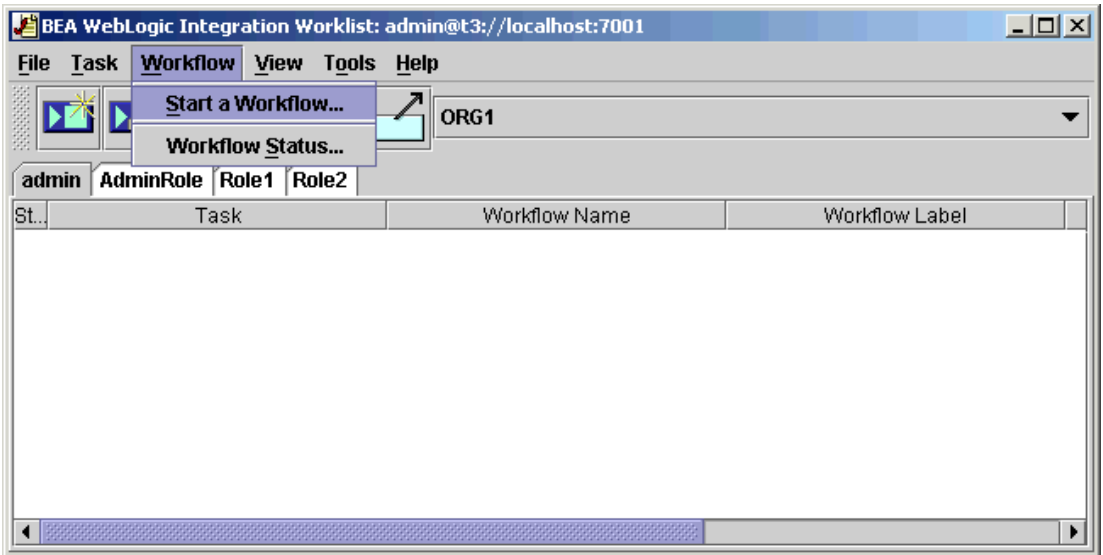
b. Click OK.

You can now initiate the workflow from the BEA Worklist.

11. Select BEA WebLogic E-Business Platform→WebLogic Integration→Worklist.

12. Log on to Worklist.

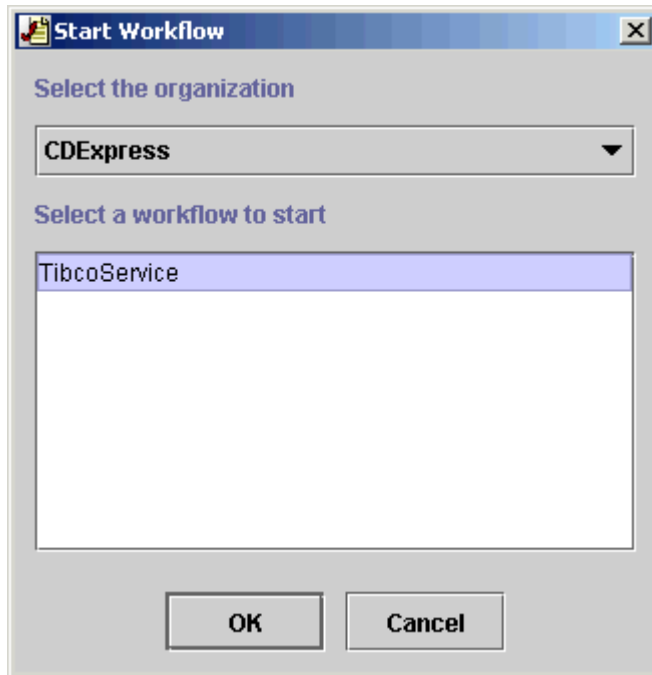
Figure 4-20 WebLogic Integration Worklist Window



13. Choose Workflow→Start a Workflow.

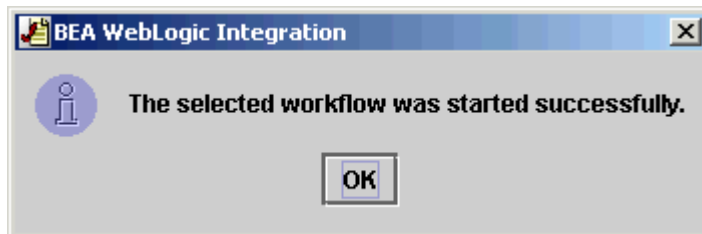
The Start Workflow dialog box appears.

Figure 4-21 Start Workflow Dialog Box



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

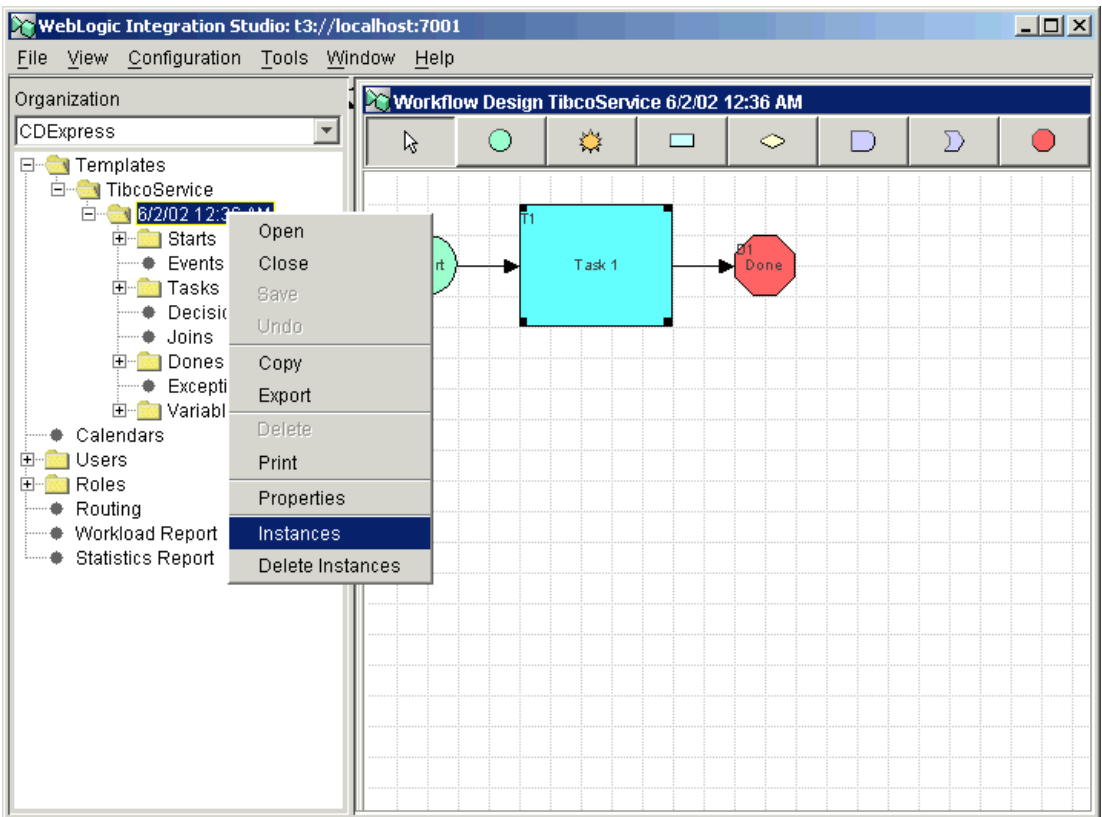
Figure 4-22 BEA WebLogic Integration - Successful Workflow Message



You receive a message indicating that your workflow has been started successfully.

4 *Creating and Configuring a Service Adapter*

Figure 4-23 WebLogic Integration Studio - Selecting Instances

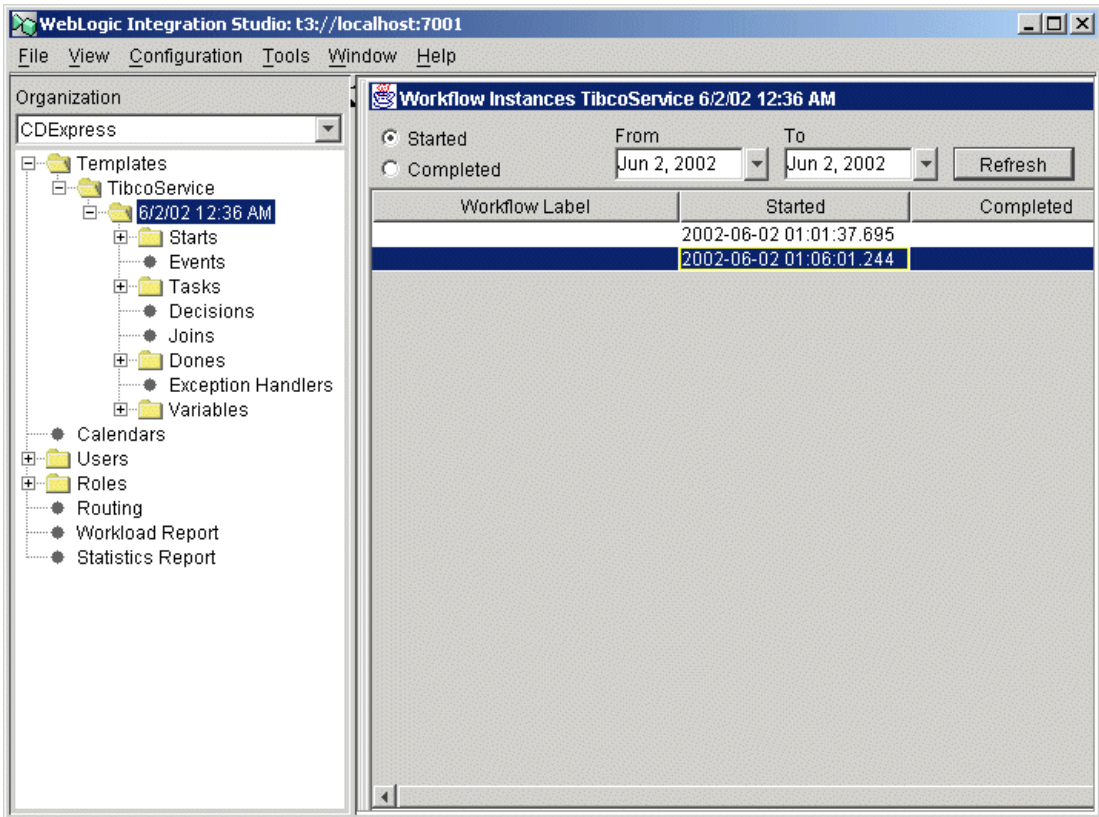


14. Right-click the service definition folder and select Instances.

15. Select Started.

16. Click Refresh.

A list of workflows appears in the right pane.

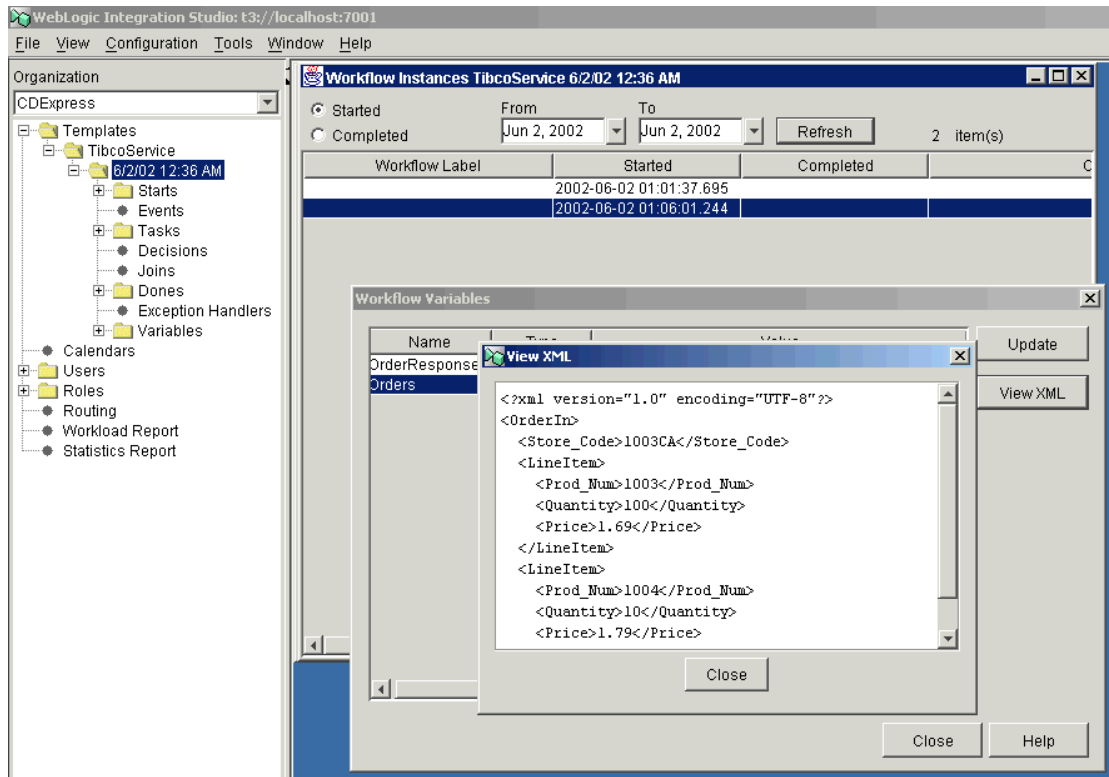
Figure 4-24 WebLogic Integration Studio - Workflow Instances Refreshed

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

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

The Workflow Variables window opens.

Figure 4-25 WebLogic Integration Studio - View XML Window



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

5 Transforming Document Formats

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

- [Kinds of Transformation](#)
- [Creating and Testing Transformations](#)

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 are two transformation phases available to an event adapter sending a message from an Enterprise Information System (EIS) to a workflow. You can invoke either phase, or both phases in sequence:

- Non-XML formats to XML.
- XML to XML.

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

- XML to XML.
- XML to non-XML formats.

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

Non-XML to XML Transformation

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

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

- Excel, which converts documents in Excel format to XML.
- MFL (Message Format Language), which uses BEA routines.

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

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

- XCH, a general conversion system which uses dictionaries (implemented as .dic files) and transformation templates (implemented as .xch files).

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

For example, the `Order_excel` event specifies the Excel transform for the non-XML to XML phase. Note that it also specifies two transforms—XCH and XSLT—for the XML to XML phase, which is described in [“XML to XML Transformations” on page 4](#).

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

The screenshot shows the 'Add Event' window in Microsoft Internet Explorer. The browser's address bar displays `http://edantk01:7001/BEA_TIBCO_1_0_Web/display.jsp`. The window title is 'Add Event - Microsoft Internet Explorer'. The main content area is titled 'Add Event' and contains the following elements:

- Navigation Sidebar:**
 - Configure Connection
 - Administration
 - Add Service
 - Add Event** (selected)
 - Deploy Application View
- Main Content Area:**
 - On this page, you add events to your application view.
 - Unique Event Name: *
 - Listener: TIBEvent
 - Configuration Table:

Daemon	EDANTK01.ibi.com
Network*	EDANTK01.ibi.com
service name	7500
Event Queue	
Send subject	
Reply subject	
Field Name	
non-XML Preparse	excel(HAS_HEADERS)
XCH Transform	Order2OrderIn.xch
XSLT Transform	OrderIn2HTML.xsl
encoding	ISO-8859-1
Polling Interval	20
 - schema:
 -

The bottom status bar shows 'Done' and 'Local intranet'.

When you specify non-XML to XML transformation in the Add Event window, use this syntax:

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

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

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

Transform Type	Parameter	Value/Description/Example
excel	HAS_HEADERS	Type/Value: string literal Description: The literal HAS_HEADERS instructs the Excel pre-parser to use the first column as the tag names for the resulting XML document. Example: excel (HAS_HEADERS)
	NO_HEADERS	Type/Value: string literal Description: The literal NO_HEADERS instructs the Excel pre-parser to generate column number tag names (for example, <col1>, <col2>) for the resulting XML document. Example: excel (NO_HEADERS)
mfl	MFL routine name	Type/Value: string Description: The name of the MFL routine. Do not enter an .mfl extension for a Message Format Language (MFL) routine; these files are not stored with an extension. Example: mfl (mfl_name)
transform	transform file name	Type/Value: string Description: The name of the transform file including the .xch extension. Example: transform (CSVtoXML.xch)

XML to XML Transformations

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

- XCH, a general conversion system which uses dictionaries (implemented as .dic files) and transformation templates (implemented as .xch files).

- **XSLT**, a language for transforming XML documents into other XML documents. It is part of XSL, the XML stylesheet language.

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

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

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

Add Event - Microsoft Internet Explorer

Address: http://edantk01:7001/BEA_TIBCO_1_0_Web/display.jsp

Add Event

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

Unique Event Name: *

Listener: TIBEvent

Daemon	<input type="text" value="EDANTK01.ibi.com"/>
Network *	<input type="text" value="EDANTK01.ibi.com"/>
service name	<input type="text" value="7500"/>
Event Queue	<input type="text"/>
Send subject	<input type="text"/>
Reply subject	<input type="text"/>
Field Name	<input type="text"/>
non-XML Preparse	<input type="text" value="excel(HAS_HEADERS)"/>
XCH Transform	<input type="text" value="Order2OrderIn.xch"/>
XSLT Transform	<input type="text" value="OrderIn2HTML.xsl"/>
encoding	<input type="text" value="ISO-8859-1"/>
Polling Interval	<input type="text" value="20"/>

schema:

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

Table 5-2 XML to XML Transform Parameters

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

XML to Non-XML

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

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

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

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

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

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

The screenshot shows the 'Add Service' window in Microsoft Internet Explorer. The browser's address bar displays a URL related to the BEA WebLogic Adapter for TIBCO. The window title is 'Add Service - Microsoft Internet Explorer'. The main content area is titled 'Add Service' and contains a sidebar with navigation links: 'Configure Connection', 'Administration', 'Add Service' (highlighted), 'Add Event', and 'Deploy Application View'. The main panel shows the 'Listener: TIBService' configuration. It includes a 'Unique Service Name' field with the value 'Order1'. Below this is a table for configuring the listener:

Transform name	ntk
Transform engine	mfl
Broker Host	EDANTK01
Broker Port	7500
Send Subject	bea.send.message
Reply Subject	
Field Name	

Below the table, there is a 'schema:' dropdown menu set to 'CUST' and an 'Add' button. The bottom status bar of the browser shows 'Done' and 'Local intranet'.

When you specify XML to non-XML transformation in the Add Service window, use this syntax:

transformType (parameter)

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

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

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

Creating and Testing Transformations

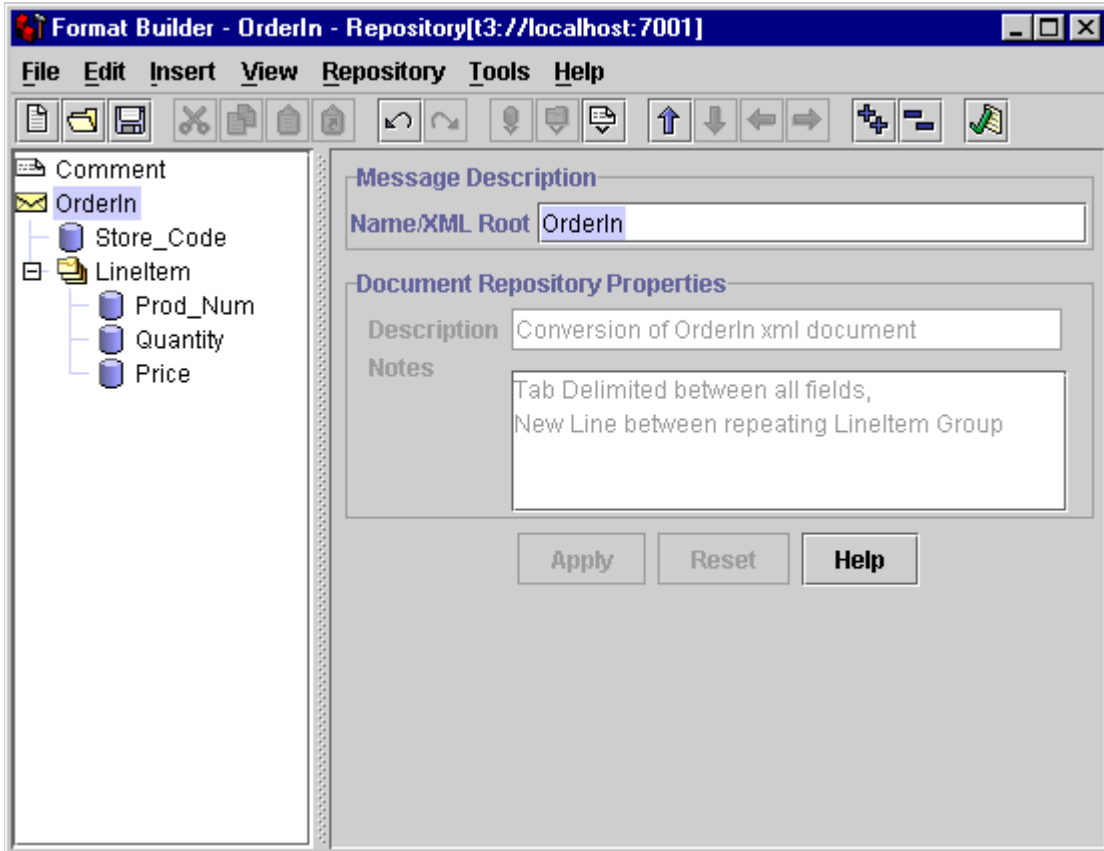
The following sections describe how to create and test transformations.

Creating MFL Transformations

This procedure explains how to construct a basic MFL transformation with the WebLogic Integration Format Builder. The MFL transformation is tested using the Application View Console.

1. Using the WebLogic Integration Format Builder, construct the OrderIn message format.

Figure 5-4 WebLogic Format Builder Main Window



2. After the message format is complete, save it to the repository.
 - a. Choose Repository→Store.
 - b. Select the desired folder in the repository.

3. When the Store Document window opens, click Store.

Figure 5-5 Store Document Window

Store Document

Current Folder: WLAI.Namespace.Root.MQSeries

There is no need to enter file extensions.

Name: OrderIn

File Type: MFL Files

Description: Conversion of OrderIn xml document

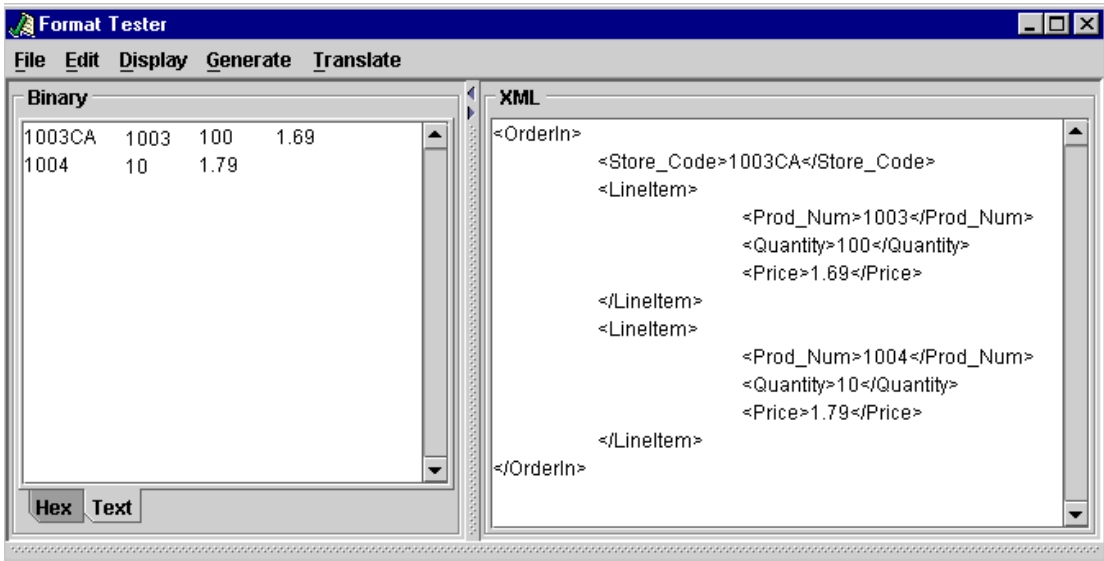
Notes: Tab Delimited between all fields,
New Line between repeating LineItem Group

Store

Cancel

Use the Format Tester to provide a preliminary confirmation of the transform.

Figure 5-6 Format Tester Window



4. In the XML pane, enter your text.
5. Choose Translate→XML to Binary.

The binary representation appears in the Binary pane.

After the MFL format definition has been stored in the repository, it may be referenced by the BEA WebLogic Adapter for TIBCO.

To test an MFT transformation, see [“Testing MFL Transformations” on page 5-12](#).

Testing MFL Transformations

This procedure explains how to test the MFL transformation using the Application View Console. In this test, the MFL transformation takes place in a service.

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

The Add Service window opens.

Figure 5-7 Add Service Window

Add Service

Application View Console WebLogic Console

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

Unique Service Name: *

Listener: TIBService

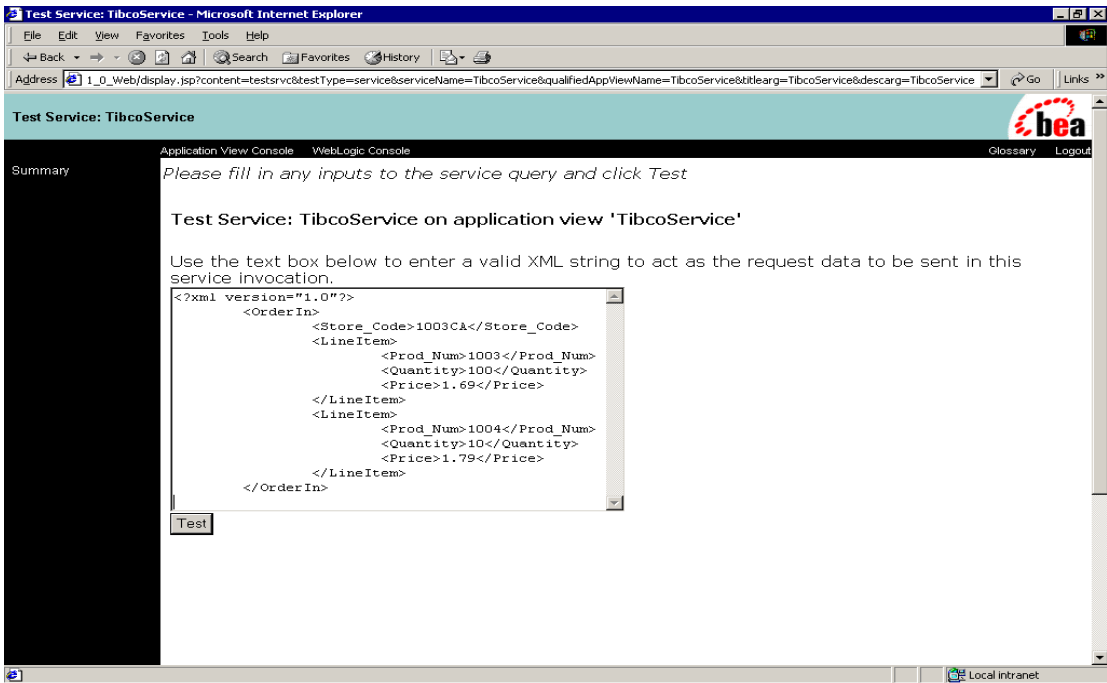
Transform name	<input type="text" value="OrderIn"/>
Transform engine	<input type="text" value="mfl"/>
Broker Host	<input type="text" value="EDANTK01"/>
Broker Port	<input type="text" value="7500"/>
Send Subject	<input type="text" value="bea.message"/>
Reply Subject	<input type="text"/>
Field Name	<input type="text"/>

schema:

- a. Update the necessary service parameters in the Add Service window.
- b. Click Add to save your parameters.

2. When the Administration window opens, click Continue.
 3. When the Deploy Application View window opens, click Deploy to deploy the new service.
 4. When the Summary window opens, click Test for the new service.
- The Test Service window opens.

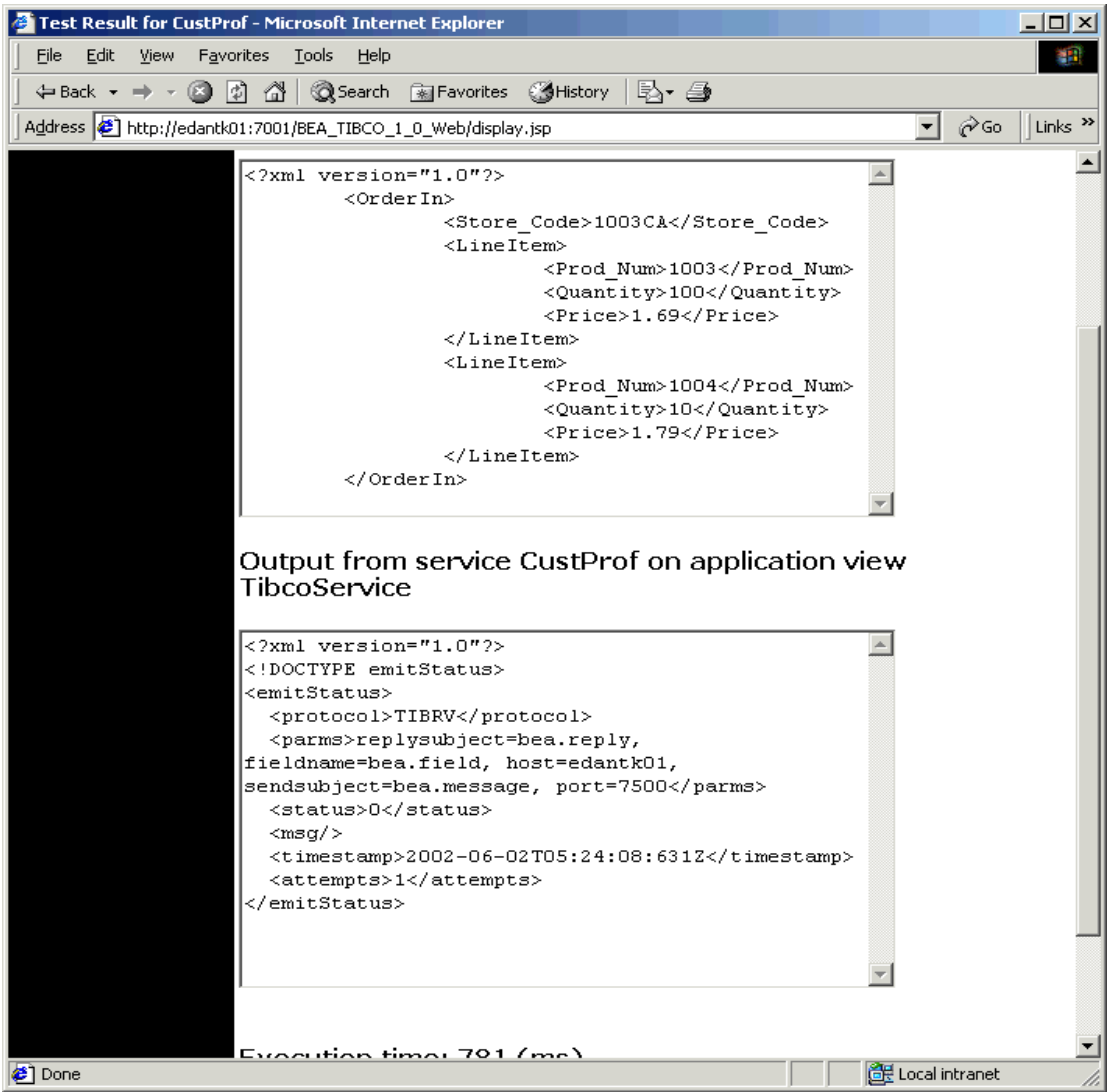
Figure 5-8 Test Service Window



5. Enter a sample XML document into the Test Service window, by either typing it or copying and pasting it.
6. Click Test to send the XML document to be transformed by the adapter into its binary equivalent as shown in [Figure 5-6, "Format Tester Window,"](#) on page 5-11.

The output shows the success or failure of the service adapter send in the lower pane of the Test Result window.

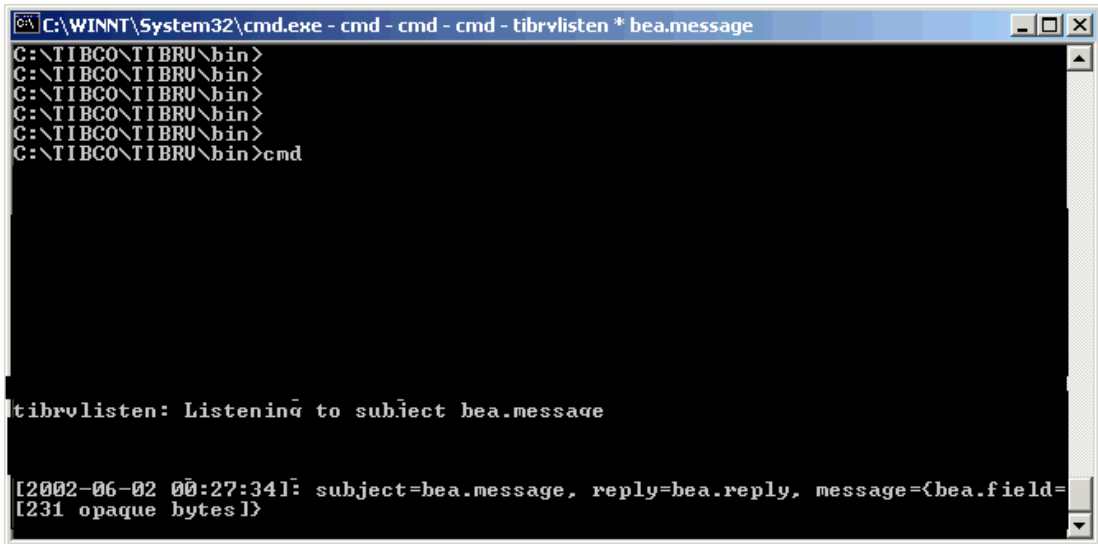
Figure 5-9 Test Result Window



The results of the transform are sent to the output queue.

7. Browse the message queues to view the new message in the TIBCO Output Queue window.

Figure 5-10 TIBCO Output Queue Window



```
C:\WINNT\System32\cmd.exe - cmd - cmd - cmd - tibrvlisten * bea.message
C:\TIBCO\TIBRU\bin>
C:\TIBCO\TIBRU\bin>
C:\TIBCO\TIBRU\bin>
C:\TIBCO\TIBRU\bin>
C:\TIBCO\TIBRU\bin>
C:\TIBCO\TIBRU\bin>cmd

tibrvlisten: Listening to subject bea.message

[2002-06-02 00:27:34]: subject=bea.message, reply=bea.reply, message={bea.field=
[231 opaque bytes]}
```


6 Creating Schema Repositories

This section addresses the schema repositories, manifests, and schemas that describe the documents entering and exiting a WebLogic Integration system. This section includes the following topics:

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

Introduction to Schemas and Repositories

You describe all the documents entering and exiting your WebLogic Integration system using W3C XML schemas. These schemas describe each event arriving to and propagating out of an event adapter, and each request sent to and each response received from a service adapter. There is one schema for each event, and two for each service (one for the request, one for the response). The schemas are usually stored in files with an `.xsd` extension.

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

BEA Adapters make use of a schema repository to store their schema information and display it in the Application View Console. The schema repository is a directory containing:

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

To work with schemas, you must know how to:

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

Naming a Schema Repository

The schema repository has a three-part naming convention:

session_base_directory\adapter_type\connection_name

Here:

session_base_directory is the schema's session base path, which represents a folder under which multiple sessions of schemas can be held.

adapter_type is the type of adapter (for example, TIBCO, MQSeries or SAP).

connection_name is a name representing a particular instance of the adapter type. For example, TIBRV01 may be a daemon on a particular host, and TIBRV02 may be another, each of these systems having different events and services relevant to them.

For example, if the session base path is `/usr/opt/bea/bse`, and the adapter type is TIBCO, and the connection name is TIBRV02, then the schema repository is the directory:

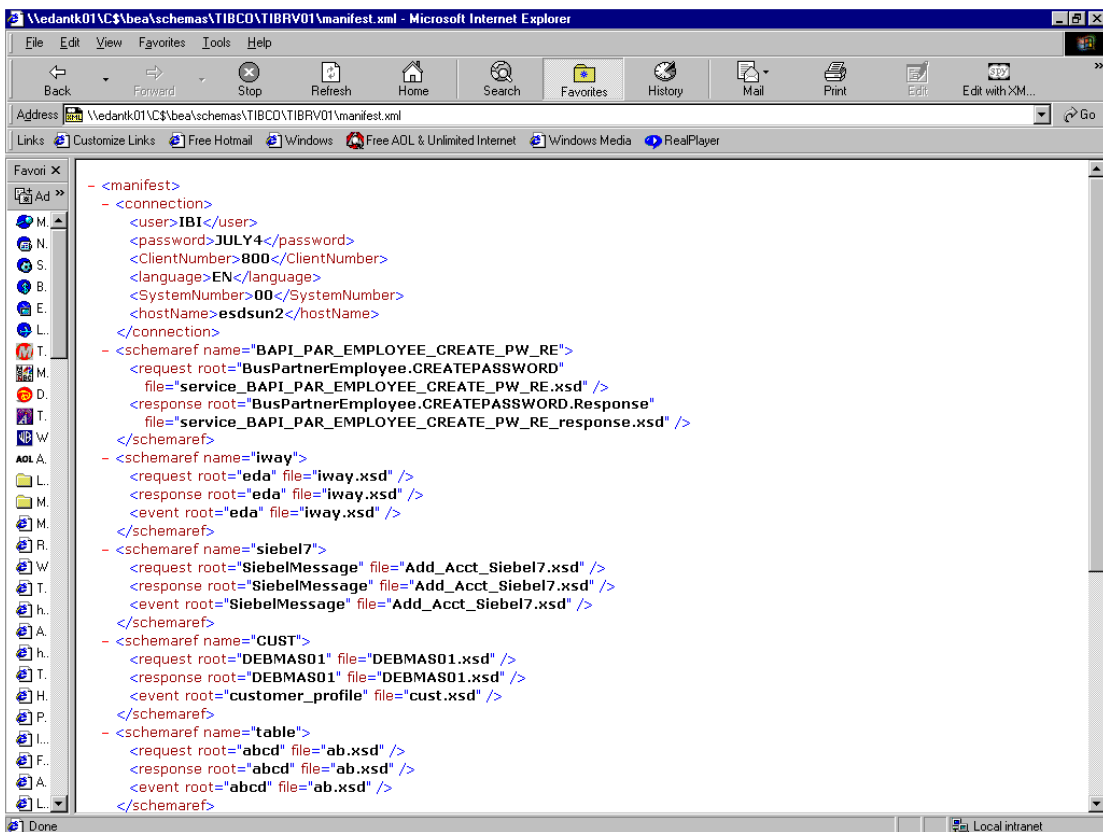
`/usr/opt/bea/bse/TIBCO/TIBRV02`

The Repository Manifest

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

The following is a sample manifest file.

Figure 6-1 Displaying a Sample Manifest File in a Web Browser



The manifest has a connection section (which is not used by the BEA WebLogic Adapter for TIBCO) and a schema reference section, named `schemaref`. The schema reference name displays in the schema drop-down list box on the Add Service and Add Event screens in the Application View Console. Each named schema reference contains three schemas, one for each of these schema types:

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

Creating a Repository Manifest

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

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

To create a manifest:

1. Create an XML file with the following structure:

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

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

```
<schemaref name="OrderIn">
  <request root="OrderIn" file="service_OrderIn_request.xsd"/>
  <response root="emitStatus" file="TibcoEmitStatus.xsd"/>
  <event root="OrderIn" file="event_OrderIn.xsd"/>
</schemaref>
```

Here, the value you assign to:

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

Creating a Schema

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

The following is an example of an instance document for the `OrderIn` event referred to in [“Creating a Repository Manifest” on page 6-4](#):

```
<?xml version="1.0"?>
<OrderIn>
  <Store_Code>1003CA</Store_Code>
  <LineItem>
    <Prod_Num>1003</Prod_Num>
    <Quantity>100</Quantity>
    <Price>1.69</Price>
  </LineItem>
  <LineItem>
    <Prod_Num>1004</Prod_Num>
    <Quantity>10</Quantity>
    <Price>1.79</Price>
  </LineItem>
</OrderIn>
```

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

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

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

  <xs:element name="LineItem">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Prod_Num"/>
        <xs:element ref="Quantity"/>
        <xs:element ref="Price"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="OrderIn">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Store_Code"/>
        <xs:element ref="LineItem"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Price">
    <xs:simpleType>
      <xs:restriction base="xs:decimal">
        <xs:enumeration value="1.69"/>
        <xs:enumeration value="1.79"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
</xs:schema>
```

```
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Prod_Num">
      <xs:simpleType>
        <xs:restriction base="xs:short">
          <xs:enumeration value="1003"/>
          <xs:enumeration value="1004"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Quantity">
      <xs:simpleType>
        <xs:restriction base="xs:byte">
          <xs:enumeration value="10"/>
          <xs:enumeration value="100"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Store_Code" type="xs:hexBinary"/>
  </xs:schema>
```


A Troubleshooting

This section lists error messages that you may encounter while using the BEA WebLogic Adapter for TIBCO. It includes the following topic:

- [Error Messages](#)

Error Messages

The following list describes error messages you may encounter while using the BEA WebLogic Adapter for TIBCO, and what you can do to resolve the errors.

BEA_TIBCO_1_0 - Can't create TIBRVEvent TIBRV protocol:
java.lang.NoClassDefFoundError: com/tibco/tibrv/TibrvException. Possibly missing jar files.

Description

This TIBCO error occurs when the class path was not updated to include the `tibrvj.jar` file.

Action

Refer to the *BEA WebLogic Adapter for TIBCO Installation and Configuration Guide* and ensure that `%TIBRV%\lib\tibrvj.jar` is added to the WebLogic Integration class path. You must stop and restart WebLogic Integration for this to take effect.

java.lang.Exception: Couldn't start EventGenerator: java.lang.IllegalStateException:
Can't get the transport connection to TIBCO Rendezvous.

Description

This adapter error occurs when the system path was not updated to include the directory containing `tibrvj.dll`

Action

Refer to the *BEA WebLogic Adapter for TIBCO Installation and Configuration Guide* and ensure that `%TIBRV%\bin` is added to the system path. You must stop and restart WebLogic Integration for this to take effect.

B Sample Integration With a SWIFT Message

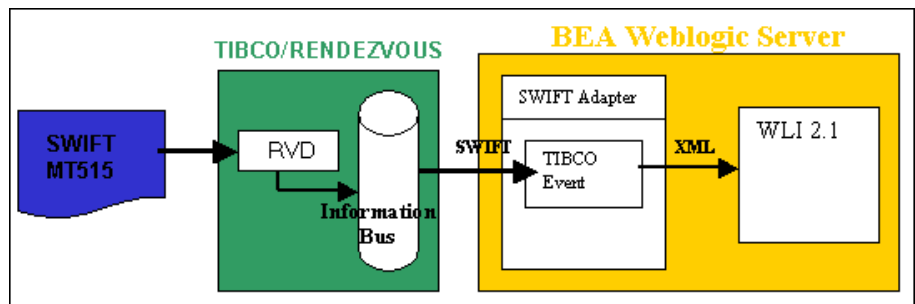
TIBCO is a widely-deployed messaging backbone in the financial services industry. Many financial services firms integrate their business processes using J2EE solutions such as BEA WebLogic Platform. This section illustrates how to send a non-XML SWIFT message between TIBCO Rendezvous and WebLogic Integration. It includes the following topics:

- [The TIBCO-Swift Integration Scenario](#)
- [Creating an Event Adapter Application View](#)
- [Configuring an Event Adapter Application View](#)
- [Validating Deployment Using WebLogic Integration Studio](#)
- [Publishing the Message Using RTTM](#)

The TIBCO-Swift Integration Scenario

In this scenario, illustrated in the following figure, you use the BEA WebLogic Adapter for SWIFT to subscribe WebLogic Integration to specified SWIFT messages, including SWIFT MT515. (MT515 is a Client Confirmation of Purchase or Sale.) The MT515 is provided only as a sample for testing the BEA WebLogic Adapter for TIBCO. The full suite of MT5## message types, dictionaries, rules, schemas, and validation rules requires the purchase of the BEA WebLogic Adapter for SWIFT.

Figure B-1 Integrating SWIFT Messages From TIBCO to WebLogic Server



You then send the SWIFT message to TIBCO Rendezvous using RTTM, a Java program provided with the adapter. The TIBCO Rendezvous daemon (**rxd**) picks up this message and routes it to the Information Bus. This is where the BEA WebLogic Adapter for TIBCO listens, picking up the SWIFT message to transform it into XML and direct it to the event adapter's router.

Creating an Event Adapter Application View

To create an event adapter application view:

1. Log on to the WebLogic Application View Console at

`//appserver-host:port/wlai`

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

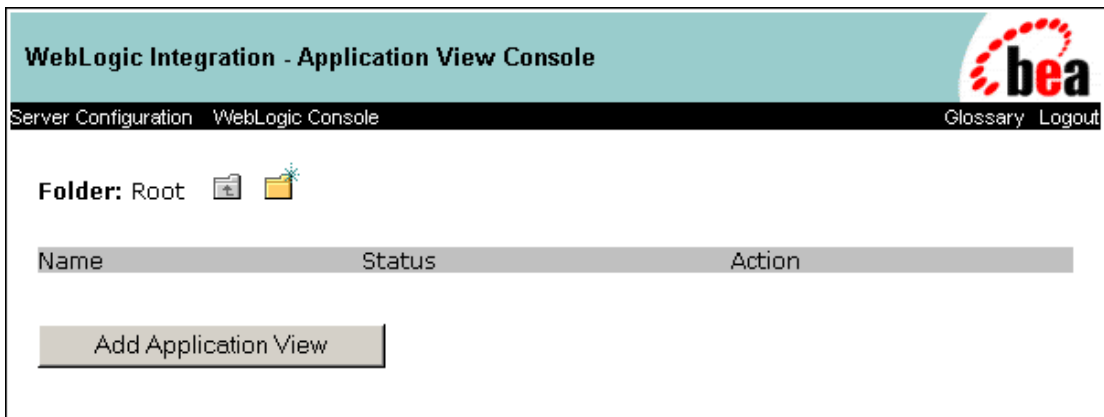
2. If prompted, enter a user name and password.

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

3. Click Login.

The WebLogic Integration Application View Console opens.

Figure B-2 Application View Console Main Window



B Sample Integration With a SWIFT Message

4. Click Add Application View.

The Define New Application View window opens.

Figure B-3 Define New Application View Window

Application View Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History Print Copy Paste CPU User

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

Define New Application View

This page allows you to define a new application view

Folder: [Root](#)

Application View Name: *

Description:

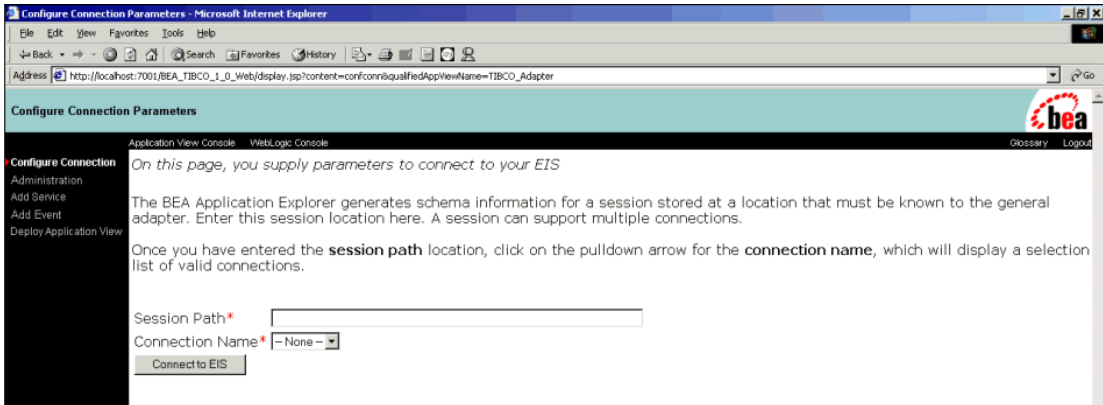
Associated Adapter:

OK Cancel

- a. Enter a name and description for the application view.
- b. Select BEA_TIBCO_1_0 from the Associated Adapter drop-down list.
- c. Click OK.

The Configure Connection Parameters window opens.

Figure B-4 Configure Connection Parameters Window



- a. Enter the name of the BEA WebLogic Adapter for TIBCO session base directory in the Session Path field.

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

For example, the session base directory might be *c:\Program Files\BEA Systems\BEA Application Explorer\sessions*, with the schema repository—containing a repository manifest and schemas—residing in *c:\Program Files\BEA Systems\BEA Application Explorer\sessions\default\TIBCO\TEST*.

For more information about schema repositories, see [Chapter 6, “Creating Schema Repositories.”](#)

- b. Select the session name—also known as the connection name—from the Connection Name drop-down list.

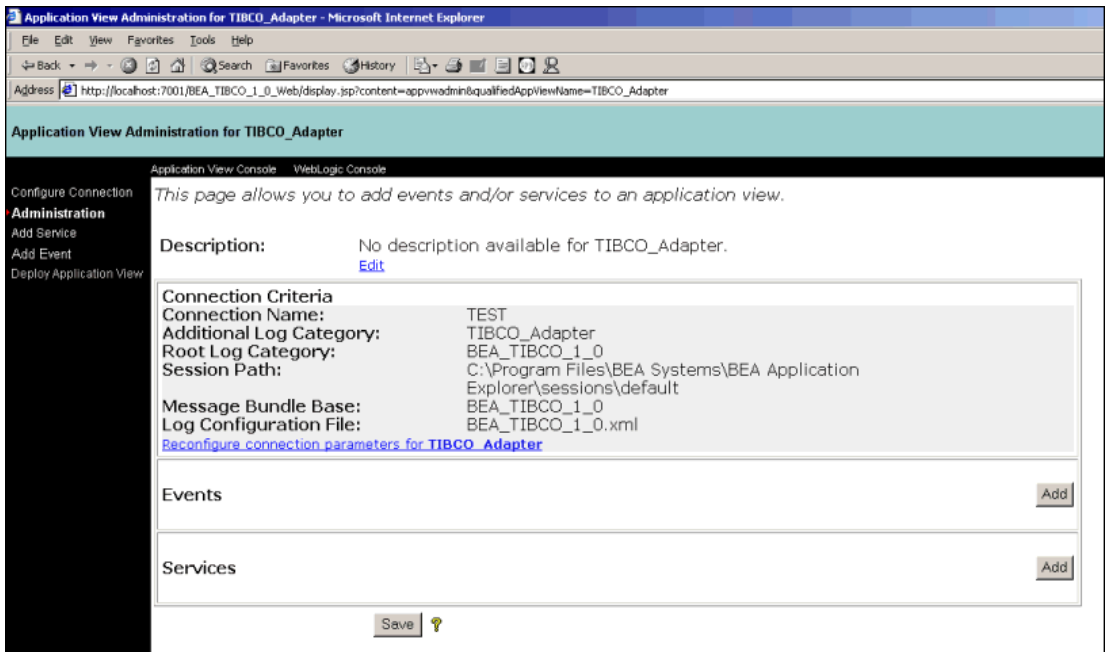
In the example in the previous step, the session name is *TEST*.

- c. Click Connect to EIS.

The Application View Administration window opens.

B Sample Integration With a SWIFT Message

Figure B-5 Application View Administration Window



5. 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 7, 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 (also known as a document). A service should be added for each event that is used by the application view.

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

1. In the Administration window of the WebLogic Application View Console, click **Add** in the Events pane.

The Add Event window opens.

B Sample Integration With a SWIFT Message

Figure B-6 Add Event Window

Add Event

Application View Console WebLogic Console

Configure Connection
Administration
Add Service
Add Event
Deploy Application View

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

Unique Event Name: *

TIBEvent

Daemon	<input type="text" value="7500"/>
Network	<input type="text" value="tsteh2000"/>
service name	<input type="text" value="7500"/>
Event Queue	<input type="text"/>
Send subject	<input type="text" value="BEA"/>
Reply subject	<input type="text"/>
Field Name	<input type="text" value="DATA"/>
non-XML Preparse	<input type="text" value="transform(MT515toXML.xch)"/>
XCH Transform	<input type="text"/>
XSLT Transform	<input type="text"/>
encoding	<input type="text" value="ISO-8859-1"/>
Polling Interval	<input type="text" value="20"/>

schema:

settings

root to transform template directory	<input type="text" value="transform/xch"/>
root to XML Style sheet directory	<input type="text" value="transform/xslt"/>

- a. Enter the event name `MyEvent` in the Unique Event Name field.

The event properties that appear correspond to TIBCO Rendezvous communication and transformation settings. The event adapter uses these settings to communicate with TIBCO Rendezvous and to process messages.

- b. Set the following parameters:

Daemon

Defines the TIBCO Rendezvous daemon with which to establish a connection. You can specify the default, which is the port number: 7500.

Network

Defines the IP address at which TIBCO Rendezvous resides.

service name

Defines the UDP service to use whenever TIBCO Rendezvous conveys messages. The service name can be a name or a port number. You can specify the default, which is a port: 7500.

Send subject

The subject that identifies the message for which the event adapter is listening. Type *BEA*.

Field Name

The custom field name to use for the message field. Type *DATA*.

non-XML Parse

Identifies the transform type for the non-XML to XML transformation phase. Type `transform(MT515toXML.xch)` to specify an XCH transform from SWIFT MT515 to XML.

schema

Lists the events in the schema repository.

Select SWIFTMT515, which is the schema resulting from the transformation (SWIFT MT515 to XML).

These pre-defined transforms (and validations) are supplied with the BEA WebLogic Adapter for SWIFT.

A sample is provided for testing purposes. For information about accessing these, see “Creating Schema Repositories” in the *BEA WebLogic Adapter for SWIFT User Guide*.

- c. Click Add after you have set these parameters.

The Administration window opens.

B Sample Integration With a SWIFT Message

Figure B-7 Application View Administration Window

The screenshot shows a web browser window titled "Application View Administration for TIBCO_Adapter - Microsoft Internet Explorer". The address bar shows the URL: `http://localhost:7001/BEA_TIBCO_1_0_Web/display.jsp?content=appvwadmin`. The page has a teal header bar with the title "Application View Administration for TIBCO_Adapter". Below the header, there are two tabs: "Application View Console" and "WebLogic Console". The main content area has a left sidebar with a navigation menu containing: "Configure Connection", "Administration" (highlighted with a red arrow), "Add Service", "Add Event", and "Deploy Application View". The main content area displays the following information:

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

Description: No description available for TIBCO_Adapter. [Edit](#)

Connection Criteria

Connection Name:	TEST
Additional Log Category:	TIBCO_Adapter
Root Log Category:	BEA_TIBCO_1_0
Session Path:	C:\Program Files\BEA Systems\BEA Application Explorer\sessions\default
Log Configuration File:	BEA_TIBCO_1_0.xml
Message Bundle Base:	BEA_TIBCO_1_0

[Reconfigure connection parameters for TIBCO_Adapter](#)

Events [Add](#)

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

Services [Add](#)

At the bottom of the main content area, there are two buttons: "Continue" and "Save" with a question mark icon.

2. Click Continue.

The Deploy Application View window opens.

Figure B-8 Deploy Application View Window

The screenshot shows a web browser window titled "Deploy Application View TIBCO_Adapter to Server - Microsoft Internet Explorer". The address bar shows "http://localhost:7001/BEA_TIBCO_1_0_Web/display.jsp". The main content area is titled "Deploy Application View TIBCO_Adapter to Server" and contains the following sections:

- Application View Console** and **WebLogic Console** tabs.
- Configure Connection** link.
- Administration** link.
- Add Service** link.
- Add Event** link.
- Deploy Application View** link (highlighted in red).

The main content area contains the following configuration options:

- Required Event Parameters**: Event Router URL* (http://localhost:7001/BEA_TIBCO_1_0_EventRouter/EventRc).
- Connection Pool Parameters**: Use these parameters to configure the connection pool used by this application view.
 - Minimum Pool Size*: 1
 - Maximum Pool Size*: 10
 - Target Fraction of Maximum Pool Size*: 0.7
 - Allow Pool to Shrink?: ☒
- Log Configuration**: Set the log verbosity level for this application view. (Log warnings, errors, and audit messages)
- Configure Security**: [Restrict Access to TIBCO_Adapter using J2EE Security](#)

At the bottom, there are buttons for **Deploy**, **Save**, and a checkbox for **Deploy persistently?** (checked).

- Modify event parameters, connection pool parameters, log configuration, and security as necessary.
- Click Deploy to save and deploy the event adapter.

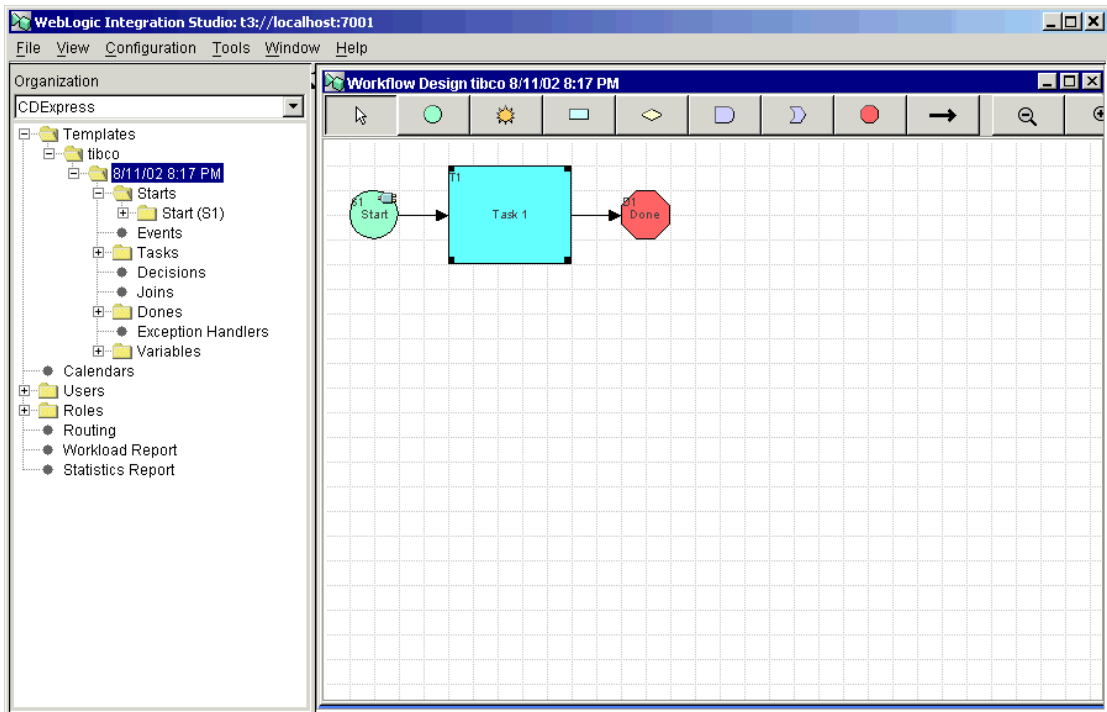
To validate the deployment, see “Validating Deployment Using WebLogic Integration Studio” on page 12.

Validating Deployment Using WebLogic Integration Studio

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

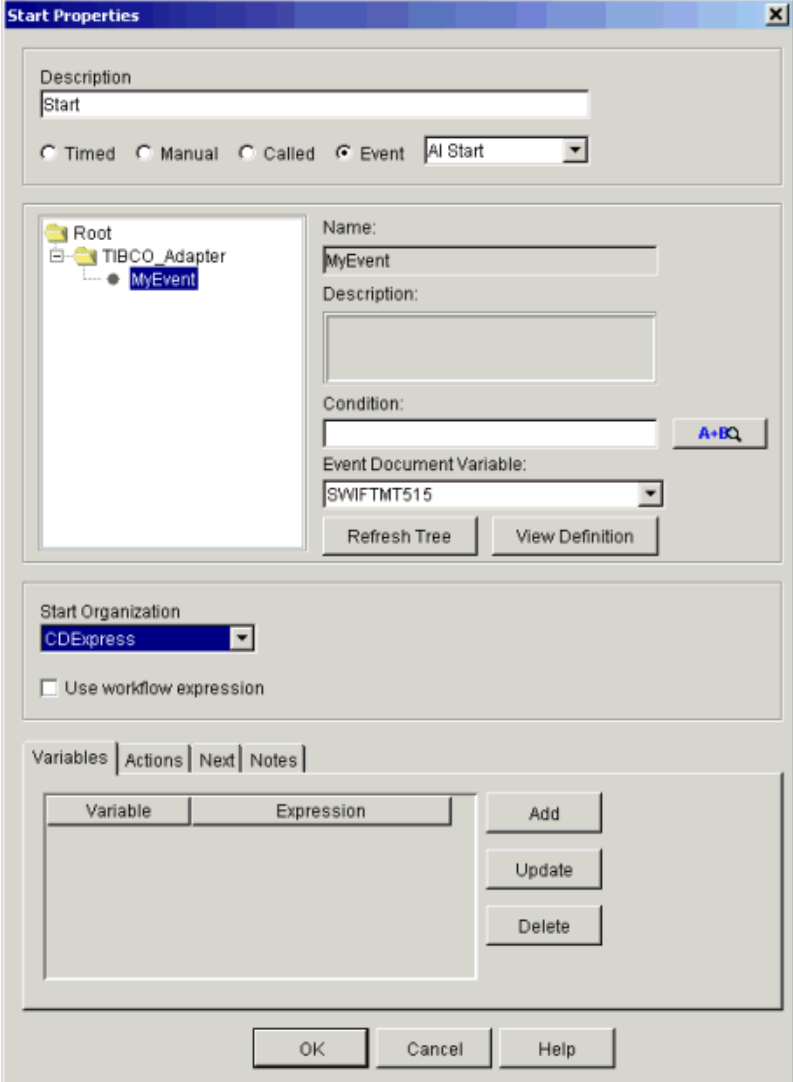
1. Start the WebLogic Integration Studio.
2. Log on to the WebLogic Integration Studio.
3. In the Organization panel, select an organization to create a new workflow template.
4. Right-click Templates and choose Create Template.
5. When the Template Properties window opens, enter a name for your workflow in the Name field.
6. Click OK.

Figure B-9 WebLogic Integration Studio



7. Right-click the new template and choose Create Template Definition.
The template opens in WebLogic Integration Studio.
8. Right-click the Start node and choose Properties (not illustrated).
The Start Properties dialog box opens.

Figure B-10 Start Properties Dialog Box



The **Start Properties** dialog box is used to configure the properties of a start event. It includes fields for Description, Name, Description, Condition, and Event Document Variable. It also features a tree view for selecting the start organization and a section for defining variables and expressions.

Description: Start

Event Type: ☐ Timed ☐ Manual ☐ Called ☒ Event **AI Start**

Event Explorer:

- Root
 - TIBCO_Adapter
 - MyEvent

Name: MyEvent

Description:

Condition:

Event Document Variable: SWIFTMT515

Start Organization: CDEExpress

☐ Use workflow expression

Variables | Actions | Next | Notes

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

Buttons: Add, Update, Delete, OK, Cancel, Help

- Select Event→AI Start.
- In the event explorer, browse the application view folders and select the application view that corresponds to the event adapter.

- c. Open the event adapter and select the MyEvent event. (You defined MyEvent earlier, in [“Configuring an Event Adapter Application View” on page B-7.](#))
- d. Select New from the Event Document Variable drop-down list.
The Variable Properties window opens (not illustrated).
 - a. Type a name for the new variable in the Event Document Variable field.
 - b. Select XML from the Type drop-down list.
 - c. Check the Input and Output Parameter options.
 - d. Click OK.
9. Right-click the template in WebLogic Integration Studio’s left panel and choose Save.
10. Right-click the event definition folder and choose Properties.
The Template Definition window opens (not illustrated).
 - a. Select Active.
 - b. Click OK.

You may now initiate events from your Enterprise Information System.

To publish a SWIFT MT515 message, see [“Publishing the Message Using RTTM” on page B-15](#)

Publishing the Message Using RTTM

You use the RTTM (Real-Time Trade Matching) Java tool to publish a SWIFT MT515 message with the message subject `BEA` and with a field named `DATA` on a specified TIBCO Rendezvous subject (BEA). You create a field named `DATA` to hold the message string. (You defined the message subject and field earlier, in [“Configuring an Event Adapter Application View” on page B-7.](#))

You can find instructions for using RTTM in `readme.txt` in `bea_tibco_samples.zip`, which is provided with the adapter.

B Sample Integration With a SWIFT Message

1. To run RTTM, add `tibrvj.jar` to your class path (it resides with TIBCO Rendezvous).
2. Enter the following command at the DOS prompt:

```
java -cp rttm.jar TestDesktop
```


RTTM's Business Participant Node window opens.

Figure B-11 RTTM: Business Participant Node

The screenshot shows the 'Business Participant Node' window with a 'New Instruction' dialog box open. The dialog box contains the following fields and values:

New Instruction	
Senders Reference	00000012
Master Reference Number	0000000013
Function	NEWM
Preparation Date	20020101121200
Buy/Sell Indicator	BUYI
Trade Date	20020101121200
BUYER	BEA
SELLER	WVAY
Settlement Date	20020101
Deal Price - Percentage	100
Trade Type	BUYI
Instruct Indicator	INST
Payment Indicator	APMT
Quantity	100
CUSIP	US/NY
Narrative	demo
SWIFT Message	{1:F01BANKGB11AXX0001000002}{2:05150112990731BANKGB11AXX00010000039907310248N}{3:{108:123456}}{4:16R:GENL:20C::SEIME#00000012:23G:NEWM:98C::PREP#20020101121200:22F::TRTR/GSCC/BUYI:16R:LINK:20C::MAST#0000000013:16S:LINK:16S:GENL:16R:CONFDET:98C::TRAD#20020101121200:98A::SETT#20020101}

Buttons at the bottom: View the Message, Send to RTTM

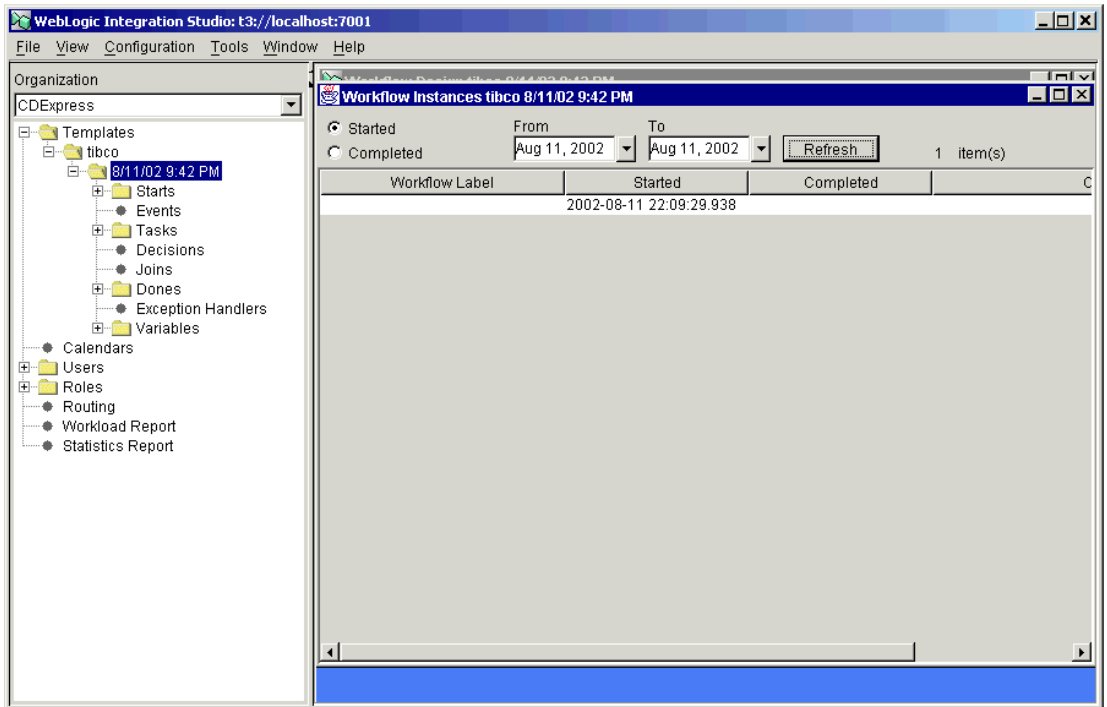
3. Enter values for the fields. You can use the sample values shown in the previous figure.

Note: Be sure to use the format `YYYYMMDDHHMMSS` for the date/time fields Preparation Date and Trade Date. Use the format `YYYYMMDD` for the date field Settlement Date. Include slashes preceding the CUSIP field's component values.

4. Click View the Message to view the message contents in the SWIFT Message field.
5. Click Send to RTTM to send the message to TIBCO Rendezvous.
6. Return to WebLogic Integration Studio.
7. To track the execution of your workflow, right-click the event definition folder and choose Instances.

The Workflow Instances for your event definition appear.

Figure B-12 WebLogic Integration Studio - Workflow Instances



- a. Select Started.
- b. Click Refresh.

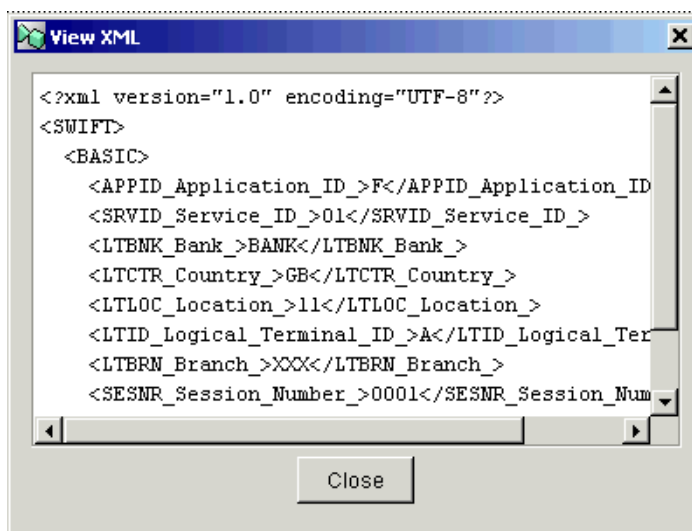
A list of started workflows appears.

8. Right-click the instance of the workflow and select Variables.

The Workflow Variables window opens.

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

Figure B-13 View XML Window



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

