



BEA WebLogic Adapter for Oracle E-Business Suite
User's Guide
Version 5 Release 5

DN3501348.0205

February 11, 2005

EDA, EDA/SQL, FIDEL, FOCCALC, FOCUS, FOCUS Fusion, FOCUS Vision, Hospital-Trac, Information Builders, the Information Builders logo, Parlay, PC/FOCUS, SmartMart, SmartMode, SNAPPack, TableTalk, WALDO, Web390, WebFOCUS and WorldMART are registered trademarks, and iWay and iWay Software are trademarks of Information Builders, Inc.

Due to the nature of this material, this document refers to numerous hardware and software products by their trademarks. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2005 BEA Systems, Inc. All Rights Reserved.

Preface

This document describes the BEA WebLogic Adapter for Oracle E-Business Suite and how to use it for developing events and services for integration with other EIS applications.

How This Manual Is Organized

The following table lists the titles and numbers of the chapters and appendixes for this manual with a brief description of the contents of each chapter or appendix.

Chapter/Appendix		Content
1	Introducing the BEA WebLogic Adapter for Oracle E-Business Suite	Introduces the BEA WebLogic Adapter for Oracle E-Business Suite and describes its key features.
2	Creating XML Schemas or Business Services for Oracle E-Business Suite	Describes how to create schemas and business services for Oracle E-Business Suite interface tables using Servlet Application Explorer.
3	Listening for Oracle E-Business Suite Events	Describes how set up ports and channels to listen for Oracle E-Business Suite events using Servlet Application Explorer.
4	Using Web Services Policy-Based Security	Describes how to configure Web services policy-based security.
5	Management and Monitoring	Describes how to use the management and monitoring tools provided by iBSE and JCA.
6	Troubleshooting	Provides information and procedures to correct common issues with the adapter and Application Explorer.
A	Using Application Explorer in BEA WebLogic Workshop to Create XML Schemas and Web Services	Describes how to use Application Explorer running in BEA WebLogic Workshop to view metadata, generate schemas, and create business services (Web services).
B	Using Application Explorer in BEA WebLogic Workshop for Event Handling	Describes how to use Application Explorer running in BEA WebLogic Workshop to listen for events in an Oracle E-Business Suite database.

Chapter/Appendix		Content
C	Open Interfaces for Oracle Release Apps 11i	Describes the agents that constitute Oracle functional components.
D	XML Business Functionality in the Adapter	Describes the business functionality of the XML supported by the BEA WebLogic Adapter for Oracle E-Business Suite.

Documentation Conventions

The following table lists and describes the conventions that apply throughout this manual.

Convention	Description
THIS TYPEFACE or <i>this typeface</i>	Denotes syntax that you must enter exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
<i>this typeface</i>	Represents a placeholder (or variable) in a text paragraph, a cross-reference, or an important term.
this typeface	Highlights a file name or command in a text paragraph that must be lowercase.
<i>this typeface</i>	Indicates a button, menu item, or dialog box option you can click or select.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices; type one of them, not the braces.
	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (...).
.	Indicates that there are (or could be) intervening or additional commands.

Contact Us!

Your feedback on the BEA WebLogic Adapter for Oracle E-Business Suite 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 Oracle E-Business Suite documentation.

In your e-mail message, please indicate that you are using the documentation for BEA WebLogic Adapter for Oracle E-Business Suite and the version of the documentation.

If you have any questions about this version of BEA WebLogic Adapter for Oracle E-Business Suite, or if you have problems using the BEA WebLogic Adapter for Oracle E-Business Suite, 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

Help Us to Serve You Better

To help our consultants answer your questions effectively, please be prepared to provide specifications and sample files and to answer questions about errors and problems.

The following tables list the specifications our consultants require.

Platform	
Operating System	
OS Version	
Product List	
Adapters	
Adapter Deployment	For example, JCA, Integration Business Services Engine
Container Version	

The following table lists components. Specify the version in the column provided.

Component	Version
Adapter	
EIS (DBMS/APP)	
HOTFIX / Service Pack	

The following table lists the types of Application Explorer. Specify the version (and platform, if different than listed previously) in the columns provided.

Application Explorer Type	Version	Platform
Swing		
Servlet		
ASP		

In the following table, specify the JVM version and vendor in the columns provided.

Version	Vendor

The following table lists additional questions to help us serve you better.

Request/Question	Error/Problem Details or Information
Provide usage scenarios or summarize the application that produces the problem.	
Did this happen previously?	
Can you reproduce this problem consistently?	
Any change in the application environment: software configuration, EIS/ database configuration, application, and so forth?	
Under what circumstance does the problem <i>not</i> occur?	
Describe the steps to reproduce the problem.	
Describe the problem .	
Specify the error message(s).	

The following table lists error/problem files that might be applicable.

XML schema
XML instances
Other input documents (transformation)
Error screen shots
Error output files

XML schema
Trace and log files
Log transaction

Contents

1. Introducing the BEA WebLogic Adapter for Oracle E-Business Suite	1-1
About the BEA WebLogic Adapter for Oracle E-Business Suite	1-2
Key Features of the BEA WebLogic Adapter for Oracle E-Business Suite	1-2
Integration Using OAGIS XML and EDI Documents	1-3
Integration Using Oracle Interface Tables	1-3
Deployment Options for the BEA WebLogic Adapter for Oracle E-Business Suite	1-4
Deployment Information Roadmap	1-4
The Integration Business Services Engine	1-5
The Enterprise Connector for J2EE Connector Architecture	1-6
2. Creating XML Schemas or Business Services for Oracle E-Business Suite	2-1
Starting Application Explorer	2-2
Opening a Connection to Oracle E-Business Suite	2-3
Creating a New Target	2-4
Connecting to a Target	2-10
Closing or Deleting a Target	2-13
Modifying a Target	2-14
Browsing Interface Table Metadata	2-15
Generating a Schema	2-17
Schema Location	2-20
Request and Response Document Formats	2-20
Interface Table Service Request Document Format	2-21
Concurrent Program Service Request Document Format	2-22
Interface Table Service Response Document Format	2-26
Concurrent Program Service Response Document Format	2-26
Generating a Business Service for an Interface Table	2-27
Generating WSDL From a Web Service	2-36
Identity Propagation	2-41
Running a Submit Request	2-42
Installing the Submit Request and Running the Concurrent.ora Script	2-43
Generating Schema for the Submit Request	2-44
Adding a Service for the Submit Request	2-44
Creating a Request Document	2-49
Testing the Configuration for the Submit Request	2-50
3. Listening for Oracle E-Business Suite Events	3-1
Understanding Event Functionality	3-2
Creating an Event Port	3-2
Editing or Deleting an Event Port	3-20

Creating a Channel	3-21
The Post Query Parameter Operators	3-28
Choosing a Listening Technique	3-29
Standard Event Processing With Row Tracking	3-30
Standard Event Processing With Row Removal	3-33
Trigger-based Event Processing	3-35
4. Using Web Services Policy-Based Security	4-1
iWay Business Services Policy-Based Security	4-2
Configuring iWay Business Services Policy-Based Security	4-3
5. Management and Monitoring	5-1
Managing and Monitoring Services and Events Using iBSE	5-2
Managing and Monitoring Services and Events Using the JCA Test Tool	5-16
Setting Engine Log Levels	5-21
Configuring Connection Pool Sizes	5-22
Migrating Repositories	5-23
File Repositories	5-23
iBSE Repositories	5-23
JCA Repositories	5-28
Migrating Event Handling Configurations	5-28
Exporting or Importing Targets	5-32
Retrieving or Updating Web Service Method Connection Information	5-36
Starting or Stopping a Channel Programmatically	5-40
6. Troubleshooting	6-1
Troubleshooting	6-2
Integration Business Services Engine Error Messages	6-3
General Error Handling in iBSE	6-3
Adapter-Specific Error Handling	6-4
BEA WebLogic Adapter for Oracle E-Business Suite Invalid SOAP Request	6-4
Failure to Connect to Oracle Applications	6-5
Invalid SOAP Request	6-5
Empty Result From an Adapter Request	6-6
A. Using Application Explorer in BEA WebLogic Workshop to Create XML Schemas and Web Services	A-1
Starting Application Explorer in BEA WebLogic Workshop	A-2
Creating a New Configuration	A-3
Connecting to Oracle E-Business Suite	A-5
Creating and Connecting to a Target	A-5
Managing a Target	A-12
Creating an XML Schema	A-15

Creating an Integration Business Service	A-17
Exporting WSDL for Use in BEA WebLogic Workshop Workflows	A-23
Identity Propagation	A-24
Adding a Control for a Resource in BEA WebLogic Workshop	A-25
Adding a Web Service Control to a BEA WebLogic Workshop Application	A-25
Adding an Extensible CCI Control to a BEA WebLogic Workshop Application	A-26
Overview	A-26
Using the Extensible CCI Control	A-33
B. Using Application Explorer in BEA WebLogic Workshop for Event Handling ...	B-1
Starting Application Explorer in BEA WebLogic Workshop	B-2
Understanding Event Functionality	B-3
Creating an Event Port	B-3
Modifying an Event Port	B-21
Creating a Channel	B-23
Modifying a Channel	B-30
Deploying Components in a Clustered BEA WebLogic Environment	B-33
C. Open Interfaces for Oracle Release Apps 11i	C-1
Oracle Financial	C-2
Oracle Manufacturing and Distribution	C-3
D. XML Business Functionality in the Adapter	D-1
BEA WebLogic Adapter for Oracle E-Business Suite and Oracle Functionality	D-2

Contents

XML Business Functionality Descriptions	D-3
Post Journal	D-3
Confirm	D-3
Process PO	D-4
Acknowledge PO	D-4
Load Receivable	D-5
Load Payable	D-5
Sync Customer	D-6
Sync Supplier	D-6
Load LdgrBudget	D-6
Get PO and Show PO	D-6
Receive PO	D-7
Get Prodorder	D-7
Transfer Item	D-8
Get Issueinfo and Show_Issueinfo	D-8
Confirm Issue	D-9
Issue MiscItem and Receive MiscItem	D-9
Get Countinfo and Show Countinfo	D-9
Update Invcnt	D-10
Get Credit, Show Credit, and Update_credit	D-10
Change Status	D-10
Load Projacctg	D-10
Sync Projinfo	D-11
Get Wipconfirm and Show Wipconfirm	D-11
Update Wipconfirm	D-11
Get PickList	D-11
List PickList	D-12
Show PickList	D-12
Update PickList	D-12
Get Personnel and Show Personnel	D-13
Update Persontime	D-13
Sync Field	D-13
Sync Personnel	D-13
Sync Wrkschedule	D-14
Sync Mfgtlcode	D-14
Sync COA	D-14
Sync Exchngrate	D-15
Add Requisitn	D-15
Change Requisitn	D-15
Cancel Requisitn	D-16
Get Requisitn and Show Requisitn	D-16
Getlist Requisitn	D-16

List Requisitn	D-17
Getlist PO	D-17
List PO	D-18
Add PO	D-18
Change PO	D-19
Cancel PO	D-19
Update Delivery	D-19
Update Inspection	D-20
Sync Item	D-20
Sync Sitelevel	D-20
Get Prodavil	D-21
Show Prodavil	D-21
Update ProductReq	D-22
Create Prodorder	D-22
Cancel Prodreq	D-22
Sync Inventory	D-23
Get BOM and Show BOM	D-23
Sync BOM	D-24
Allocate Activity	D-24
Load Matchdoc	D-24
Update Matchok and Update Matchfail	D-25
Load_PLInvoice and Load Payable	D-26
Get Matchdoc and Show Matchdoc	D-26
Getlist Picklist	D-26
Update Inventory	D-27
Getlist Countinfo and List Countinfo	D-27
Cancel Prodorder	D-28
Sync Salesorder	D-28
Add Salesorder	D-28
Cancel Salesorder	D-29
Change Salesorder	D-29
Get Salesorder and Show Salesorder	D-29
Getlist Salesorder and List Salesorder	D-30
Sync PO	D-30
Sync Routing	D-31
Get Routing and Show Routing	D-31
Getlist Routing and List Routing	D-31
Getlist BOM and List BOM	D-32
Get Item and Show Item	D-32
Getlist Item and List Item	D-33
Getlist Prodorder and List Prodorder	D-33
Sync Prodorder	D-34

Contents

Sync Engchgor dr	D-34
Get Engchgor dr and Show Engchordr	D-34
Confirm Engchordr	D-35
Sync Dspchlist	D-35
Get Dspchlist and Show Dspchlist	D-36
Update Dspchlist	D-36
Sync Maintorder	D-36
Create Maintorder	D-37
Update Maintorder	D-37
Cancel Maintorder	D-37
Get Maintorder and Show Maintorder	D-37
Getlist Maintorder List Maintorder	D-38
Sync UOMGroup	D-38
Get UOMGroup and Show UOMGroup	D-39
Getlist UOMGroup and List UOMGroup	D-39
Sync Catalog	D-40
Get Catalog and Show Catalog	D-41
Sync Itemclass	D-41
Get Itemclass and Show Itemclass	D-42
Sync ITEMXREF	D-43
Get ITEMXREF and Show ITEMXREF	D-44
Sync Pricelist	D-44
Get Pricelist and Show Pricelist	D-45
Sync Itemspecs	D-46
Get Itemspecs and Show Itemspecs	D-47
Sync RFQ	D-48
Add RFQ	D-48
Change RFQ	D-49
Cancel RFQ	D-49
Respond RFQ	D-49
Getlist RFQ and List RFQ	D-50
Get RFQ and Show RFQ	D-51
Sync Quote	D-51
Add Quote	D-51
Change Quote	D-52
Cancel Quote	D-52
Respond Quote	D-53
Getlist Quote and List Quote	D-53
Get Quote and Show Quote	D-54
Show Shipment	D-54
Sync Planschd	D-55
Get Planschd and Show Planschd	D-55

Sync Seqsched (Sequence Schedule) D-56

Get Seqsched (Sequence Schedule) and Show Seqsched (Sequence Schedule) D-56

Sync Shipschd (Shipment Schedule) D-57

Get Shipschd (Shipment Schedule) and Show Shipschd (Shipment Schedule) D-57

Process Invoice D-58

Process WIPSPLIT D-58

Process WIPMERGE D-59

Process WIPRECOVER D-59

Get WIPSTATUS and Show WIPSTATUS D-60

Process WIPMOVE D-60

Allocate Resource D-61

Get Customer and Show Customer D-61

Get Supplier and Show Supplier D-62

Sync Ecatalog D-62

Get ECATALOG and Show ECATALOG D-63

Sync Invoice D-63

Get Invoice and Show Invoice D-64

Get Consumption and Show Consumption D-64

Create Requisitn D-65

Getlist LDGRACTUAL and List LDGRACTUAL D-65

Get LDGRACTUAL and Show LDGRACTUAL D-66

Acknowledge Delivery D-66

Receive Delivery D-67

Get Delivery and Show Delivery D-67

Getlist Delivery and List Delivery D-68

Contents

CHAPTER 1

Introducing the BEA WebLogic Adapter for Oracle E-Business Suite

Topics:

- About the BEA WebLogic Adapter for Oracle E-Business Suite
- Integration Using OAGIS XML and EDI Documents
- Integration Using Oracle Interface Tables

The BEA WebLogic Adapter for Oracle E-Business Suite enables you to reuse your existing Oracle E-Business Suite procedures and applications with other applications, a key to building a successful e-business or integrated enterprise.

This section introduces the BEA WebLogic Adapter for Oracle E-Business Suite and describes its key features.

About the BEA WebLogic Adapter for Oracle E-Business Suite

The BEA WebLogic Adapter for Oracle E-Business Suite provides simple open standard access to Oracle E-Business Suite through Oracle E-Business Suite open interface tables, published APIs, the iProcurement Connector, the e-Commerce Gateway, and the XML Gateway. No recoding or modifications of the Oracle E-Business Suite system are required. The adapter can integrate documents based on open standards such as Open Application Group Integration Specification (OAGIS), Electronic Data Interchange (EDI), specialized user defined formats, and traditional XML.

In addition, the BEA WebLogic Adapter for Oracle E-Business Suite enables you to fully integrate an Oracle E-Business Suite system with other enterprise resources, such as a Database Management System (DBMS) that has a JDBC™ 2.0–compliant driver, e-mail system, HTTP protocol–based server, or FTP server.

From the adapter you also can connect to a host of enterprise integration systems (EIS), including popular enterprise resource planning (ERP), supply chain management (SCM), and customer relationship management (CRM) applications.

The BEA WebLogic Adapter for Oracle E-Business Suite comprises several execution agents that enable integration with other systems by functioning in one of two ways:

- **Requests for Oracle inbound processing.** The adapter sends requests to the Oracle E-Business Suite system through Oracle interface tables and Oracle published APIs.
- **Listeners for Oracle outbound events.** The adapter listens for application-based table activity.

Key Features of the BEA WebLogic Adapter for Oracle E-Business Suite

Key features of the BEA WebLogic Adapter for Oracle E-Business Suite include:

- Requests that communicate with Oracle published interface tables and Oracle published APIs.
- Asynchronous as well as synchronous processing.
- Bidirectional message/request processing.
- XML-based requests and responses.
- Back-end error propagation through the RDBMS Table Listener.
- Oracle business logic and functionality that ensures accuracy and maintainability.
- Management and monitoring of services and events.
- Component deployment in a clustered BEA WebLogic environment.

- Application Explorer, which enables you to:
 - Browse Oracle E-Business Suite metadata.
 - Generate XML schemas for service requests.

Integration Using OAGIS XML and EDI Documents

The BEA WebLogic Adapter for Oracle E-Business Suite enables you to take advantage of Oracle supplied gateways into and out of Oracle applications. You can use the BEA WebLogic Adapter for Oracle E-Business Suite in conjunction with Oracle e-Business adapters to transfer event information using OAGIS-formatted XML (Open Applications Group Specification) and EDI (Electronic Data Interchange) documents. Then, these documents are processed by the Oracle iProcurement Connector or the e-Commerce Gateway, respectively. This type of integration is ideal if your organization currently uses these technologies because it enables quick integration of additional processes through an existing system.

The Oracle Advanced Queuing (AQ) system passes and retrieves OAGIS-formatted XML documents to and from the iProcurement Connector. EDI documents are passed and retrieved by reading files from specified directories.

The external system generates the following types of event documents:

- Documents that do not support the OAGIS/EDI formats.
- Documents that do support the OAGIS/EDI formats.

These documents may have extensions or require data enrichment or cleansing.

You perform data enrichment or cleansing before loading the data into Oracle E-Business Suite.

Integration Using Oracle Interface Tables

The BEA WebLogic Adapter for Oracle E-Business Suite enables you to import data into Oracle E-Business Suite using specific Oracle-supplied interface tables. Instead of transforming input documents into standards-based XML structures, the request document is transformed to match the appropriate Oracle-specific interface table.

Interface tables are one mechanism Oracle E-Business Suite uses to expose its business processing. Data is never loaded directly into Oracle E-Business Suite base tables. Instead, data loads into interface tables, and Oracle-supplied concurrent programs move data from interface to base tables, ensuring that all business logic and processing is handled through Oracle-authored components.

Deployment Options for the BEA WebLogic Adapter for Oracle E-Business Suite

The BEA WebLogic Adapter for Oracle E-Business Suite works in conjunction with Servlet Application Explorer with either:

- Integration Business Services Engine (iBSE)
- Enterprise Connector for J2EE™ Connector Architecture (JCA)

Application Explorer, used to access Oracle E-Business Suite metadata and create Web services and events, can be configured to work in a Web services environment in conjunction with iBSE or the Enterprise Connector for JCA. When working in a JCA environment, the connector uses the Common Client Interface (CCI) to provide fast integration services using an adapter instead of Web services.

Both iBSE and the Connector for JCA are deployed to your application environment with Application Explorer and the adapters.

Deployment Information Roadmap

The following table lists the location of deployment information for the BEA WebLogic Adapter for Oracle E-Business Suite. A description of the Integration Business Services Engine (iBSE) and the Enterprise Connector for J2EE Connector Architecture (JCA) follow the table.

Deployed Component	For more information, see
Application Explorer	Chapters 2 and 3 of this guide Appendix A and Appendix B of this guide when using Application Explorer inside BEA WebLogic Workshop <i>BEA WebLogic ERP Adapter Installation and Configuration</i> manual
Integration Business Services Engine (iBSE)	<i>BEA WebLogic ERP Adapter Installation and Configuration</i> manual
Enterprise Connector for J2EE Connector Architecture (JCA)	<i>BEA WebLogic ERP Adapter Installation and Configuration</i> manual

The following versions of Application Explorer are available when deploying the adapter with BEA WebLogic Server:

- **Servlet.** Deployed as a Web application on BEA WebLogic Server, this version is accessible through a Web browser. In addition, the servlet Application Explorer can be used with Integration Business Services Engine (iBSE) and Enterprise Connector for J2EE Connector Architecture (JCA). For more information, see the following chapters:
 - Chapter 2, *Creating XML Schemas or Business Services for Oracle E-Business Suite*
 - Chapter 3, *Listening for Oracle E-Business Suite Events*
- **Integrated Java Swing.** Tightly integrated within the BEA WebLogic toolset, the integrated Java Swing Application Explorer can be accessed directly from BEA WebLogic WorkShop, where WSDL (Web Services Description Language) files generated from Integration Business Services and XML schemas can be shared as resources within a BEA WebLogic WorkShop application. For more information, see Appendix A, *Using Application Explorer in BEA WebLogic Workshop to Create XML Schemas and Web Services* and Appendix B, *Using Application Explorer in BEA WebLogic Workshop for Event Handling*

Note: To use Application Explorer within BEA WebLogic WorkShop, you must deploy the Integration Business Services Engine (iBSE). For more information, see the *BEA WebLogic ERP Adapter Installation and Configuration* manual.

The Integration Business Services Engine

The Integration Business Services Engine (iBSE) exposes—as Web services—enterprise assets that are accessible from adapters regardless of the programming language or the particular operating system.

iBSE simplifies the creation and execution of Web services when running:

- Custom and legacy applications
- Database queries and stored procedures
- Packaged applications
- Terminal emulation and screen-based systems
- Transactional systems

Web services is a distributed programming architecture that solves Enterprise Application Integration (EAI) hurdles that other programming models cannot. It enables programs to communicate with one another using a text-based but platform and language independent message format called XML.

Coupled with a platform and language independent messaging protocol called SOAP (Simple Object Access Protocol), XML enables application development and integration by assembling previously built components from multiple Web services.

The Enterprise Connector for J2EE Connector Architecture

The Enterprise Connector for J2EE Connector Architecture (JCA) enables developers of JCA-compliant applications to deploy adapters as JCA resources.

The Connector for JCA is distributed as a standard Resource Adapter Archive (RAR) for deployment to the application server. Thus, the connector can be used in systems that are non-compliant, although services such as pooled connections are not available.

CHAPTER 2

Creating XML Schemas or Business Services for Oracle E-Business Suite

Topics:

- Starting Application Explorer
- Opening a Connection to Oracle E-Business Suite
- Closing or Deleting a Target
- Modifying a Target
- Browsing Interface Table Metadata
- Generating a Schema
- Request and Response Document Formats
- Generating a Business Service for an Interface Table
- Running a Submit Request

This section describes how to create XML schemas and business services using Servlet Application Explorer . Although this section describes the Java™ servlet implementation of Application Explorer, other implementations provide the same functionality by means of similar graphical user interfaces.

For information on using Application Explorer in BEA WebLogic Workshop, see Appendix A, *Using Application Explorer in BEA WebLogic Workshop to Create XML Schemas and Web Services*.

Starting Application Explorer

You can use Servlet Application Explorer to:

- Generate XML schemas that define request and response documents for your Oracle interface tables. You can use these schemas when you create request documents and when you develop logic for processing response documents.
- View metadata that describes your interface tables. You can use this metadata when you create request documents and when you develop logic for processing response documents.
- Create business services (also known as Web services) for your interface tables.

Procedure: How to Start Application Explorer

To start Application Explorer:

1. Start BEA WebLogic Server.

On Windows:

Click the *Start* menu and select *Programs, BEA WebLogic Platform 8.1, User Projects*, then your domain, and then click *Start WLS Server*.

On UNIX or from a command line:

Type the following at the prompt

```
BEA_HOME\user_projects\domains\DOMAIN_NAME\startWebLogic.cmd
```

where:

```
BEA_HOME
```

Is the directory in which BEA WebLogic is installed.

```
DOMAIN_NAME
```

Is the domain you are using.

2. After BEA WebLogic Server is running, enter the following URL in your browser window

```
http://hostname:port/iwae/index.html
```

where:

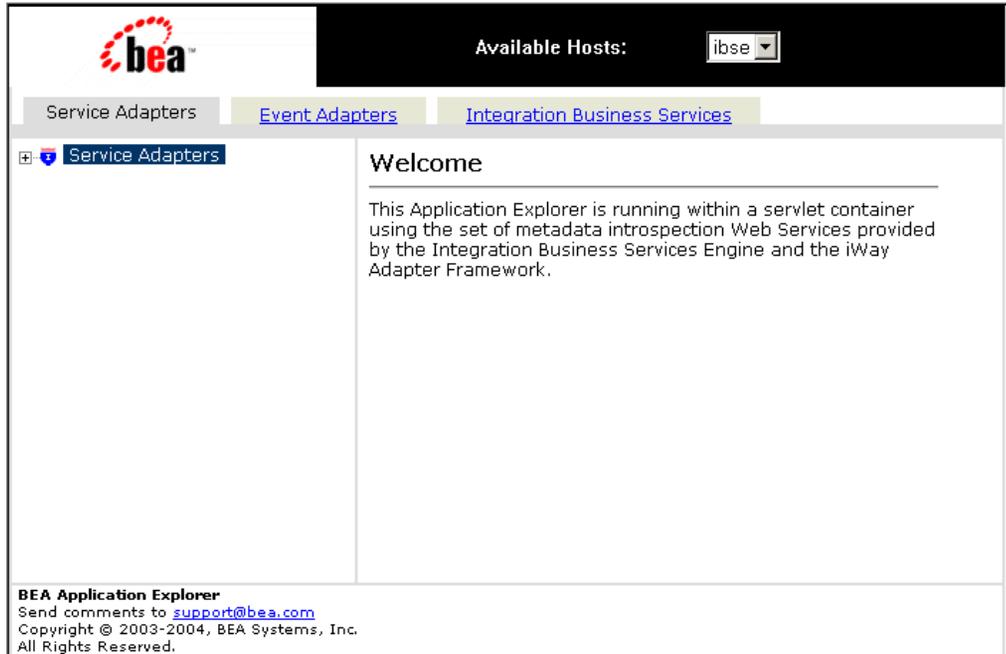
```
hostname
```

Is the name of the host on which your application server is running.

```
port
```

Is the number of the port on which the application server is operating. The port for the default domain is 7001.

After you start Application Explorer, the following image shows the Welcome window displaying the Service Adapters, Event Adapters, and iWay Business Services tabs. The Service Adapters node is highlighted in the left pane and welcome information appears in the right pane.



The Available Hosts drop-down list in the top frame determines which JCA Connector and Servlet iBSE instances you can access. For information about adding instances, see the *BEA WebLogic ERP Adapter Installation and Configuration* manual.

3. To display the list of available adapters, expand the *Service Adapters* node in the left pane.

You are now ready to add new targets to Oracle E-Business Suite.

Opening a Connection to Oracle E-Business Suite

To browse and work with Oracle E-Business Suite tables, you must create a target for Oracle E-Business Suite. This target serves as your connection point. You must establish a connection to Oracle every time you start Servlet Application Explorer or after you disconnect from the system.

You can connect to Oracle E-Business Suite by:

- Creating a new target.

- Connecting to an existing target.

The left pane displays the application systems supported by Application Explorer, which are based on the adapters you installed and are licensed to use.

Creating a New Target

A target serves as the connection point to your Enterprise Information System (EIS) and is automatically saved after you create it. To connect to Oracle E-Business Suite for the first time, you must create a new target.

Procedure: How to Create a New Target

To create a new target:

1. In the left pane, expand the *Service Adapters* node and select the *Oracle Applications* node.

The following image shows the list of supported adapters in the left pane and information about the selected adapter on the right.



2. In the right pane, move the pointer over *Operations* and select *Define a new target*.

The Add a new ORACLE APPLICATIONS target pane opens on the right, as shown in the following image. This pane provides fields to define the new target.

Add a new ORACLE APPLICATIONS target

Targets represent configured connections to instances of backend systems. Choose a name and description for the new target that you wish to create.

Target Name:

Description:

Target Type:

- a. In the Target Name field, type a descriptive name for the target, for example, OracleTarget.
- b. In the Description field, type a brief description of the connection.
- c. From the Target Type drop-down list, select the type of Oracle client to which you want to connect.

The options are Thin Client or OCI Client.

3. Click *Next*.

The Set connection info pane opens on the right.

If you selected a Thin Client target type, the Set connection info pane appears as in the following image. This image shows the fields for the Thin Client connection parameters and the active buttons.

The image shows a dialog box titled "Set connection info". It contains the following fields and buttons:

- User: [text input field]
- Password: [text input field]
- Host: [text input field]
- Port: [text input field]
- SID: [text input field]
- Concurrent TNS Name: [text input field]
- Batch size: [text input field]
- Buttons: Help (disabled), < Back, Finish, Cancel

Note: The Oracle E-Business Suite connection parameters are consistent with those found in your Oracle E-Business Suite system. For more information on parameter values that are specific to your Oracle E-Business Suite configuration, consult your Oracle E-Business Suite system administrator.

The following table lists and describes each connection parameter for a Thin Client target.

Parameter	Description
User	The Oracle database user ID to access the Oracle database underlying the Oracle E-Business Suite system. The user ID must have database access to the interface tables being accessed.
Password	The password associated with the specified user ID.
Host	The name of the server on which the Oracle E-Business Suite database instance resides.

Parameter	Description
Port	The port number on which the database is listening.
SID	A unique name of the database service, chosen by the database administrator or the person who installed Oracle E-Business Suite.
Concurrent TNS Name	<p>The logical name of the Oracle E-Business Suite database instance. (You can obtain this from the database administrator.)</p> <p>The TNS name that is found in the tnsnames.ora file that contains an entry for that database and that resides on the Oracle E-Business Suite database server. This is <i>not</i> the TNS name that is registered on the server on which the BEA WebLogic Adapter for Oracle E-Business Suite is running (unless the adapter and the Oracle database reside on the same server).</p> <p>If a TNS name does not exist, you must create it. For more information about creating a TNS name, see your Oracle administrator.</p>
Batch size	<p>The number of request document records that the adapter buffers before inserting them into an Oracle E-Business Suite table. When the adapter reaches the end of the request document, it inserts any remaining buffered records.</p> <p>The adapter tracks this batch limit separately for each table, not aggregately for all tables. For example, if you set the batch size property to 25, and the adapter accumulates 18 input records for the GL_INTERFACE table and 25 for the BUDGET_INTERFACE table, the adapter continues to hold records for GL_INTERFACE but inserts the records for BUDGET_INTERFACE.</p> <p>The batch size property enables you to optimize I/O: inserting too few records at a time increases I/O overhead, and inserting too many at a time ties up database resources.</p>

If you selected an OCI Client target type, the Set connection info pane appears as in the following image. This image shows the fields for the OCI Client connection parameters and the active buttons.

The image shows a dialog box titled "Set connection info". It contains the following fields and controls:

- User: [Text input field]
- Password: [Text input field]
- TNS Name: [Text input field]
- Release: [Dropdown menu with "8" selected]
- Concurrent TNS Name: [Text input field]

At the bottom of the dialog, there are four buttons: "Help", "< Back", "Finish", and "Cancel".

Note: The Oracle E-Business Suite connection parameters are consistent with those found in your Oracle E-Business Suite system. For more information on parameter values that are specific to your Oracle E-Business Suite configuration, consult your Oracle E-Business Suite system administrator.

The following table lists and describes each connection parameter for an OCI Client target.

Parameter	Description
User	Oracle database user ID to access the Oracle database underlying the Oracle E-Business Suite system. The user ID must have database access to the interface tables being accessed.
Password	Password associated with the specified user ID.

Parameter	Description
TNS Name	<p>Logical name of the Oracle E-Business Suite database instance. (You can obtain this from the database administrator.)</p> <p>The TNS name that is found in the tnsnames.ora file that contains an entry for that database and that <i>resides on the server running the adapter.</i></p> <p>If a TNS name does not exist, you must create it. For information about creating a TNS name, see your Oracle administrator.</p>
Release	<p>Release number of the Oracle E-Business Suite instance. Select it from the list of releases supported by the adapter.</p>
Concurrent TNS Name	<p>Logical name of the Oracle E-Business Suite database instance. (You can obtain this from the database administrator.)</p> <p>The TNS name that is found in the tnsnames.ora file that contains an entry for that database and that <i>resides on the server hosting the Oracle E-Business Suite database.</i></p> <p>If a TNS name does not exist, you must create it. For information about creating a TNS name, see your Oracle administrator.</p>

4. Enter values for the connection parameters. The Thin Client and OCI Client connection parameter values are defined in the tables found in Step 3.
5. Click *Finish*.

The new target appears in the left pane beneath the Oracle Applications node, as shown in the following image.



You are ready to connect to the application target you defined.

Connecting to a Target

You must use the targets you define to connect to an instance of Oracle E-Business Suite.

Procedure: How to Connect to a Target

To connect to a target:

1. In the left pane, expand the *Service Adapters* node, the *Oracle Applications* node, and select the target to which you want to connect.
2. In the right pane, move the pointer over *Operations* and select *Connect*.

If the target type is Thin Client, the Connect to *target name* pane opens on the right. The following image is an example of this pane for a target named OracleTarget. It shows the parameter fields (User, Password, Host, Port, SID, Concurrent TNS Name, and Batch size) with example values.

The image shows a dialog box titled "Connect to OracleTarget". It contains several input fields with the following values:

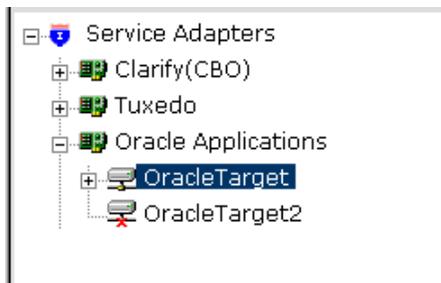
Field	Value
User:	apps
Password:	
Host:	oracle11x
Port:	1523
SID:	vis
Concurrent TNS Name:	
Batch size:	

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

If the target type is OCI Client, the Connect to *target name* pane appears on the right. The following image is an example of this pane for a target named OracleTarget2. It shows the connection parameters (User, Password, TNS Name, Release, Concurrent TNS Name) with example values.

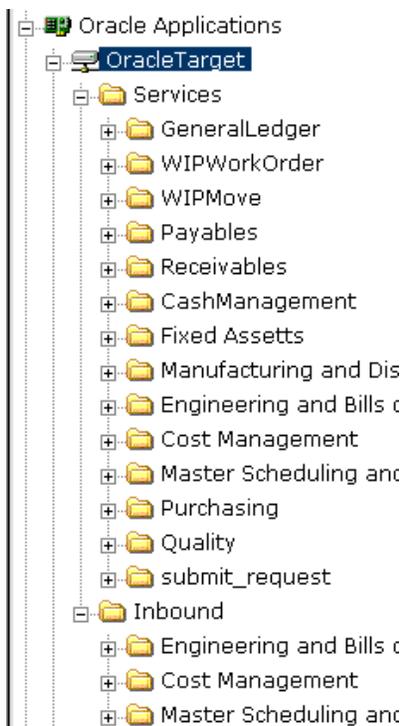
3. In the Password field, type a valid password.
4. Click *OK*.

The following image shows a target node, OracleTarget, with a plus sign next to it, indicating that it is connected.



5. Expand the target node.

The following is an image of the left pane showing an expanded target node named OracleTarget.



Closing or Deleting a Target

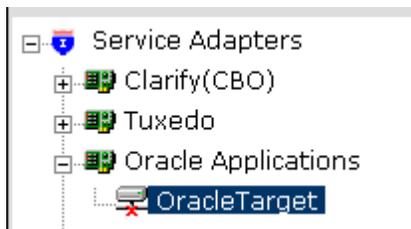
Although you can maintain multiple open connections to different application systems, we recommend that you close connections that are not in use.

Procedure: How to Disconnect From a Target

To disconnect from a target:

1. Click the target that you want to close.
2. In the right pane, move the pointer over *Operations* and select *Disconnect*.

Disconnecting from Oracle E-Business Suite drops the connection, but the target definition and its node remain accessible from the left pane. The following image shows a red x under the target, OracleTarget, to indicate that it is disconnected.



Procedure: How to Delete a Target

To delete a target from Application Explorer:

1. In the left pane, select the target you want to delete.
2. In the right pane, move the pointer over *Operations* and select *Delete*.
A confirmation dialog box opens.
3. To delete the target, click *OK*.

The target node disappears from the left pane.

Modifying a Target

After a target exists in Servlet Application Explorer, you can edit the target name, description, and connection parameters. You must disconnect from a target before you can edit it.

Procedure: How to Edit a Target

To edit a target in Application Explorer:

1. Disconnect from the target you want to edit. See *Closing or Deleting a Target* on page 2-13.
2. In the left pane, select the target you want to edit.
3. In the right pane, move the pointer over *Operations* and select *Edit*.

The Edit ORACLE APPLICATIONS target pane opens on the right. The following image is an example of the Edit ORACLE APPLICATIONS target pane showing the current connection settings for Target Name, Description, and Target Type.

Edit ORACLE APPLICATIONS target OracleTarget

Targets represent configured connections to instances of backend systems. Choose a name and description for the new target that you wish to create.

Target Name:

Description:

Target Type:

4. Modify the target information as required.
5. Click *Next*.

The Set connection info pane opens. This pane is described in *Creating a New Target* on page 2-4.

6. Edit the parameters in the Set connection info pane as required.
7. Click *Finish*.

The target information reflects the changes made through this edit.

Browsing Interface Table Metadata

Browsing metadata in Servlet Application Explorer can be useful when creating request documents.

Procedure: How to View Interface Table Metadata

To view Oracle E-Business Suite interface table metadata:

1. Connect to an Oracle E-Business Suite target, as described in *Opening a Connection to Oracle E-Business Suite* on page 2-3.

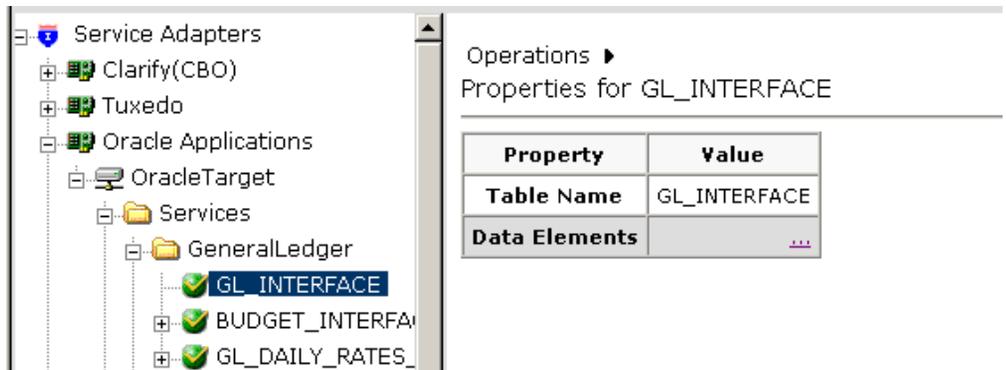
2. In the left pane, open the target node to display its modules.

The list includes all modules currently supported by Oracle E-Business Suite.

3. Expand the *Services* node and select a table whose metadata you want to view.

The user ID for this target connection must have read access to the selected table; otherwise you cannot view its metadata.

When you select a specific table on the left, the property table appears on the right. The following image shows an example of a table selected on the left and the properties of the table (property and value) listed on the right.



4. In the property table, click the browse link ([...](#)) in the Data Elements row.

A detailed table appears on the right. The following image shows an example of a detailed interface table, which includes columns for the field name, the SQL data type, and for indicating whether the parameter is required.

Details for collection property Data Elements

Field Name	SQL Type	Required
STATUS	VARCHAR	true
SET_OF_BOOKS_ID	NUMERIC	true
ACCOUNTING_DATE	DATE	true
CURRENCY_CODE	VARCHAR	true
DATE_CREATED	DATE	true
CREATED_BY	NUMERIC	true
ACTUAL_FLAG	VARCHAR	true
USER_JE_CATEGORY_NAME	VARCHAR	true
USER_JE_SOURCE_NAME	VARCHAR	true
CURRENCY_CONVERSION_DATE	DATE	false
ENCUMBRANCE_TYPE_ID	NUMERIC	false
BUDGET_VERSION_ID	NUMERIC	false
USER_CURRENCY_CONVERSION_TYPE	VARCHAR	false
CURRENCY_CONVERSION_RATE	NUMERIC	false
SEGMENT1	VARCHAR	false
SEGMENT2	VARCHAR	false
SEGMENT3	VARCHAR	false
SEGMENT4	VARCHAR	false

Use this information to determine the table(s) and fields to use when creating an XML request document or business service.

5. Scroll to the bottom of the property table and click *Close* to hide the detail rows and return to the Operations menu.

Generating a Schema

When you deploy the BEA WebLogic Adapter for Oracle E-Business Suite in the iWay Business Services Engine (iBSE) environment or an Enterprise Connector for J2EE Connector Architecture (JCA) environment, you can generate schemas that define a service request document and the corresponding response document.

For more information about request and response documents, see *Request and Response Document Formats* on page 2-20.

If you plan to deploy the BEA WebLogic Adapter for Oracle E-Business Suite in a business services environment, you are not required to generate a schema. For more information, see *Generating a Business Service for an Interface Table* on page 2-27.

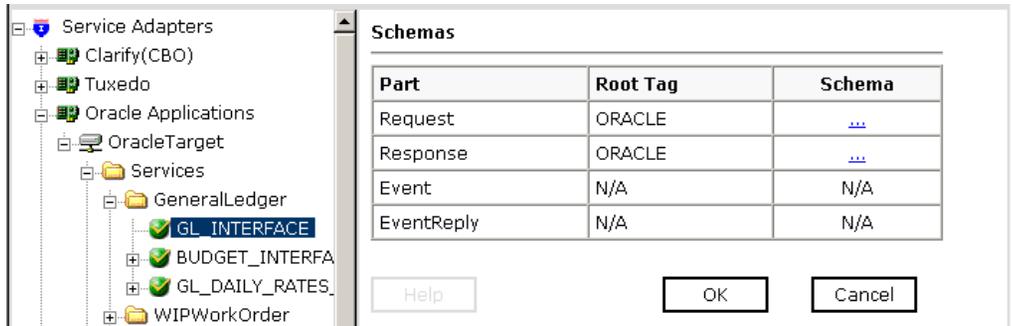
Note: During runtime, it is the responsibility of the workflow manager in BEA WebLogic to handle multiple requests with the same session ID.

Procedure: How to Generate Schema for an Interface Table

To generate a schema for an Oracle E-Business Suite interface table using Application Explorer:

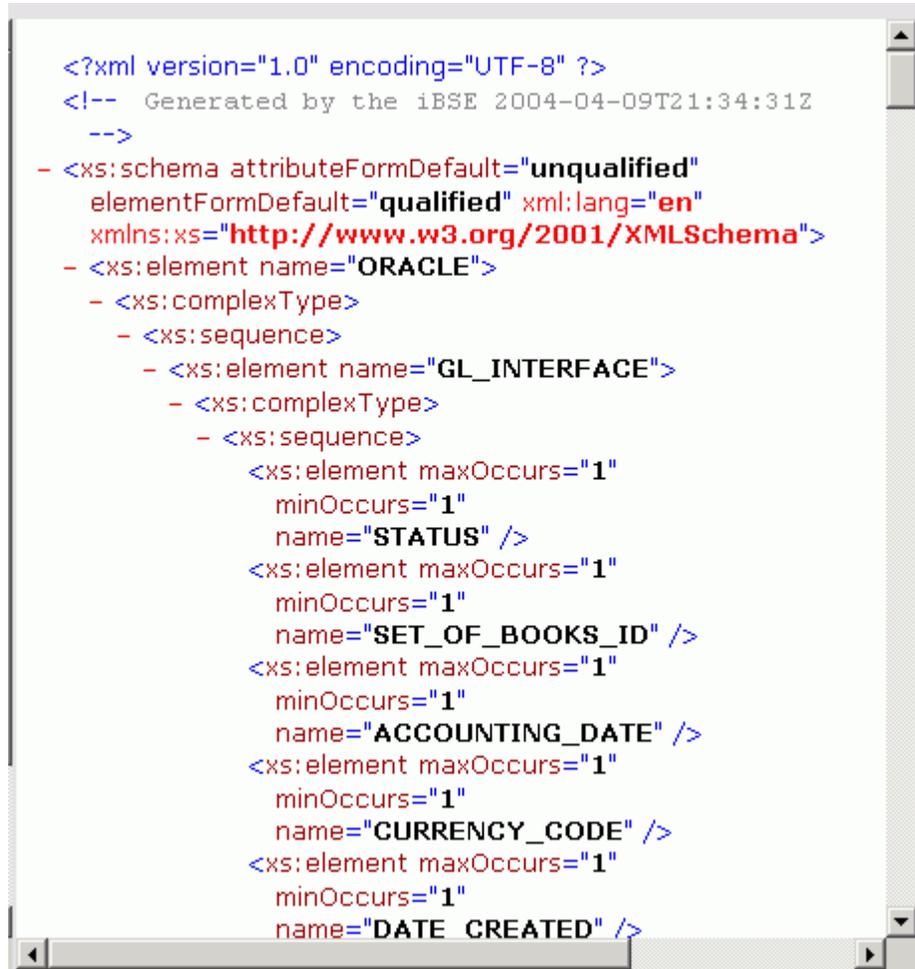
1. Connect to an Oracle E-Business Suite target, as described in *Opening a Connection to Oracle E-Business Suite* on page 2-3.
2. In the left pane, expand the target node to display its modules.
The list includes all modules currently supported by Oracle E-Business Suite.
3. Open the module containing the table for which you want to create schemas.
4. Select the table.
The user ID for this target connection must have read access to the selected table; otherwise you cannot create its schemas.
5. In the right pane, move the pointer over *Operations* and select *Generate Schema*.

The Schemas pane opens on the right, as shown in the following image, with a table listing Part, Root Tag, and Schema. The Schema column provides hyperlinks to the different schemas.



6. In the Schema column, click the browse hyperlink (...) associated with the type of schema (request or response) you want to view.

The schema appears in the right pane, as shown in the following image.



```

<?xml version="1.0" encoding="UTF-8" ?>
<!-- Generated by the iBSE 2004-04-09T21:34:31Z
-->
- <xs:schema attributeFormDefault="unqualified"
  elementFormDefault="qualified" xml:lang="en"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
- <xs:element name="ORACLE">
- <xs:complexType>
- <xs:sequence>
- <xs:element name="GL_INTERFACE">
- <xs:complexType>
- <xs:sequence>
  <xs:element maxOccurs="1"
    minOccurs="1"
    name="STATUS" />
  <xs:element maxOccurs="1"
    minOccurs="1"
    name="SET_OF_BOOKS_ID" />
  <xs:element maxOccurs="1"
    minOccurs="1"
    name="ACCOUNTING_DATE" />
  <xs:element maxOccurs="1"
    minOccurs="1"
    name="CURRENCY_CODE" />
  <xs:element maxOccurs="1"
    minOccurs="1"
    name="DATE_CREATED" />

```

- a. Right-click the right pane and select *View Source*.
 - b. Save the schema or copy and paste it to another location.
7. To return to the Schemas table, click the *Back* button of your Web browser.
 8. To return to the Operations menu, click *OK*.

Schema Location

Application Explorer stores the schemas it creates in subdirectories under the iWay home directory of the machine where it is installed. The exact location of the schemas differs depending on whether you deploy Application Explorer with an iBSE or a JCA configuration.

- **When using the adapter with an iBSE configuration**, the schemas are stored under a \schemas subdirectory of the iWay home directory, for example,

```
C:\Program Files\iway55\bea\ibse\wsdl\schemas\service\Oracle  
Applications\OracleTarget
```

where:

OracleTarget

Is the name of the connection to the Oracle Applications system that you defined in Application Explorer. Under this directory, Application Explorer creates subdirectories containing schemas.

- **When using the adapter with a JCA configuration**, the schemas are stored under a \schemas subdirectory of the iWay home directory, for example,

```
C:\Program Files\iWay55\config\base\schemas\Oracle Applications  
\OracleTarget
```

where:

OracleTarget

Is the name of the connection to the Oracle Applications system that you defined in Application Explorer. Application Explorer stores the schemas in this directory.

Request and Response Document Formats

This topic describes the format of BEA WebLogic Adapter for Oracle E-Business Suite service request and response documents. You can generate request and response document schemas using Application Explorer. Then, use a third-party XML tool to generate a request document instance from the schema.

For services, you can generate request and response documents for:

- Interface tables
- Concurrent programs

When processing a request document, the adapter performs all interface table insertions before it runs any concurrent programs (using the `submit_request` element). If an interface table insertion fails, all insertions are rolled back; otherwise, they are committed, regardless of the concurrent program result.

Interface Table Service Request Document Format

The format of an interface table service request document sent by the BEA WebLogic Adapter for Oracle E-Business Suite to Oracle is

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

<ORACLE>
  <interface_table_name>
    <column_name>data</column_name>
    . . .
    <column_name>data</column_name>
  </interface_table_name>
</ORACLE>
```

where:

interface_table_name

Identifies the interface table where data will be inserted.

column_name

Is the name of an interface table column.

data

Is the data inserted into the interface table column.

Concurrent Program Service Request Document Format

The format of a concurrent program service request document sent by the BEA WebLogic Adapter for Oracle E-Business Suite to Oracle is

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

<ORACLE>
  <submit_request>
    <resp_appl_shrtnm>responsibility_app_short_name</resp_appl_shrtnm>
    <responsibility>responsibility</responsibility>
    <username>user</username>
    <wait>wait_indicator</wait>
    <prog_appl_shrtnm>program_app_short_name </prog_appl_shrtnm>
    <program>program_name</program>
    <parm>parameter</parm>
    . . .
    <parm>parameter</parm>
  </submit_request>

  <submit_request>
    . . .
  </submit_request>
  .
  .
  .
</ORACLE>
```

where:

submit_request

Identifies a concurrent program invocation. Each concurrent program invocation requires its own *submit_request* element. One document can include multiple *submit_request* elements.

responsibility_app_short_name

Is the application short name of your responsibility. SYSADMIN is the default value.

responsibility

Is the responsibility of the application user. SYSADMIN is the default value.

user

Is the name of your Application Object Library user. This name is used to update WHO information for data that the concurrent program changes and to create the report output file name for the specific request. This parameter is different from the user ID used to log on to the Oracle database. SYSADMIN is the default value.

wait_indicator

Indicates whether the agent waits until the concurrent program finishes.

Y specifies synchronous processing: the agent waits as the system checks the status of the concurrent program every 60 seconds.

N, the default value, specifies asynchronous processing: the agent does not wait.

program_app_short_name

Is the application short name of your program. This value cannot be null.

program_name

Is the name of the concurrent program. This value cannot be null.

parameter

Is a concurrent program parameter. The parameters are identified by sequence. You can determine parameter sequence in the Oracle E-Business Suite Web Client Concurrent Program Parameters window.

Example: Interface Table Request Document for WIP_JOB_SCHEDULE_INTERFACE

The following interface table service request document inserts a record into WIP_JOB_SCHEDULE_INTERFACE.

```
<ORACLE>
  <WIP_JOB_SCHEDULE_INTERFACE>
    <LAST_UPDATE_DATE>2002-12-02</LAST_UPDATE_DATE>
    <LAST_UPDATED_BY>1001611</LAST_UPDATED_BY>
    <CREATION_DATE>2002-12-02</CREATION_DATE>
    <CREATED_BY>1001611</CREATED_BY>
    <LAST_UPDATE_LOGIN/>
    <REQUEST_ID/>
    <PROGRAM_ID/>
    <PROGRAM_APPLICATION_ID/>
    <PROGRAM_UPDATE_DATE/>
    <GROUP_ID>2</GROUP_ID>
    <SOURCE_CODE/>
    <SOURCE_LINE_ID/>
    <PROCESS_TYPE/>
    <ORGANIZATION_ID/>
    <LOAD_TYPE>1</LOAD_TYPE>
    <STATUS_TYPE>3</STATUS_TYPE>
    <OLD_STATUS_TYPE/>
    <LAST_UNIT_COMPLETION_DATE/>
    <OLD_COMPLETION_DATE/>

  <PROCESSING_WORK_DAYS/>
    <DAILY_PRODUCTION_RATE/>
    <LINE_ID/>
    <PRIMARY_ITEM_ID>155</PRIMARY_ITEM_ID>
    <BOM_REFERENCE_ID/>
    <ROUTING_REFERENCE_ID/>
    <BOM_REVISION_DATE/>
    <ROUTING_REVISION_DATE/>
    <WIP_SUPPLY_TYPE>7</WIP_SUPPLY_TYPE>
    <CLASS_CODE>Discrete</CLASS_CODE>
    <LOT_NUMBER/>
    <LOT_CONTROL_CODE/>
    <JOB_NAME/>
    <DESCRIPTION/>
    <FIRM_PLANNED_FLAG/>
    <ALTERNATE_ROUTING_DESIGNATOR/>
    <ALTERNATE_BOM_DESIGNATOR> </ALTERNATE_BOM_DESIGNATOR>
    <DEMAND_CLASS/>
    <START_QUANTITY>100</START_QUANTITY>
    <OLD_START_QUANTITY/>
    <WIP_ENTITY_ID/>
    <REPETITIVE_SCHEDULE_ID/>
```

```

<ERROR/>
<PARENT_GROUP_ID/>
<ATTRIBUTE_CATEGORY/>
<ATTRIBUTE1/>
<ATTRIBUTE2/>
<ATTRIBUTE3/>
<ATTRIBUTE4/>
<ATTRIBUTE5/>
<ATTRIBUTE6/>
<ATTRIBUTE7/>
<ATTRIBUTE8/>
<ATTRIBUTE9/>
<ATTRIBUTE10/>
<ATTRIBUTE11/>
<ATTRIBUTE12/>
<ATTRIBUTE13/>
<ATTRIBUTE14/>
<ATTRIBUTE15/>
<INTERFACE_ID/>
<LAST_UPDATED_BY_NAME/>
<CREATED_BY_NAME/>
<PROCESS_PHASE>2</PROCESS_PHASE>
<PROCESS_STATUS>1</PROCESS_STATUS>
<ORGANIZATION_CODE>M1</ORGANIZATION_CODE>
<FIRST_UNIT_START_DATE>2002-12-02</FIRST_UNIT_START_DATE>
<FIRST_UNIT_COMPLETION_DATE>2002-12-02</FIRST_UNIT_COMPLETION_DATE>
<LAST_UNIT_START_DATE>2002-12-02</LAST_UNIT_START_DATE>
<SCHEDULING_METHOD/>
<LINE_CODE/>
<PRIMARY_ITEM_SEGMENTS/>
<BOM_REFERENCE_SEGMENTS/>
<ROUTING_REFERENCE_SEGMENTS/>
<ROUTING_REVISION/>
<BOM_REVISION/>
<COMPLETION_SUBINVENTORY/>
<COMPLETION_LOCATOR_ID/>
<COMPLETION_LOCATOR_SEGMENTS/>
<SCHEDULE_GROUP_ID/>
<SCHEDULE_GROUP_NAME/>
<BUILD_SEQUENCE/>
<PROJECT_ID/>
<PROJECT_NAME/>
<TASK_ID/>
<TASK_NAME/>
<NET_QUANTITY>100</NET_QUANTITY>
<DESCRIPTIVE_FLEX_SEGMENTS/>
<PROJECT_NUMBER/>
<TASK_NUMBER/>

```

```
<PROJECT_COSTED/>
<END_ITEM_UNIT_NUMBER/>
<OVERCOMPLETION_TOLERANCE_TYPE/>
<OVERCOMPLETION_TOLERANCE_VALUE/>
<KANBAN_CARD_ID/>
<PRIORITY>2</PRIORITY>
<DUE_DATE>2002-12-02</DUE_DATE>
<ALLOW_EXPLOSION>Y</ALLOW_EXPLOSION>
<HEADER_ID/>
<DELIVERY_ID/>
<COPRODUCTS_SUPPLY/>
<DUE_DATE_PENALTY/>
<DUE_DATE_TOLERANCE/>
</WIP_JOB_SCHEDULE_INTERFACE>

</ORACLE>
```

Interface Table Service Response Document Format

The format of an interface table service response document returned to the BEA WebLogic Adapter for Oracle E-Business Suite is

```
<?xml version="1.0" encoding="UTF-8"?>
<ORACLE>
  <Processed_table>interface_table_name</Processed_table>
</ORACLE>
```

where:

interface_table_name

Is the name of the interface table that was processed.

Concurrent Program Service Response Document Format

The format of a concurrent program service response document returned to the BEA WebLogic Adapter for Oracle E-Business Suite is

```
<?xml version="1.0" encoding="UTF-8"?>
<ORACLE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="schema_location">
  <submit_response>
    <request_id>requestID</request_id>
  </submit_response>
</ORACLE>
```

where:

schema_location

Is the name and full path of the schema.

requestID

Is the request ID assigned by Oracle E-Business Suite to the request job.

Example: Response Document for WIP_JOB_SCHEDULE_INTERFACE

The following BEA WebLogic Adapter for Oracle E-Business Suite response document was returned by Oracle after inserting a record into the WIP_JOB_SCHEDULE_INTERFACE interface table.

```
<ORACLE>
  <Processed_table>WIP_JOB_SCHEDULE_INTERFACE</Processed_table>
</ORACLE>
```

Generating a Business Service for an Interface Table

You can generate a business service (also known as a Web service) for an Oracle E-Business Suite interface table. To generate a business service, you must deploy the BEA WebLogic Adapter for Oracle E-Business Suite in a business services environment using the iWay Business Services Engine (iBSE). iBSE exposes functionality as Web services and serves as a gateway to heterogeneous back-end applications and databases.

A Web service is a self-contained, modularized function that can be published and accessed across a network using open standards. It is the implementation of an interface by a component and is an executable entity. For the caller or sender, a Web service can be considered a “black box” that may require input and delivers a result. Web services can be integrated within an enterprise as well as across enterprises on any communication technology stack, whether asynchronous or synchronous, in any format.

You can make a Web service available to other services within a host server by generating WSDL (Web Services Description Language) from the Web service.

Ensure that the servlet iBSE is properly configured. For more information on installing and deploying components, see the *BEA WebLogic ERP Adapter Installation and Configuration* manual.

Note: Web services are not available in a J2EE Connector Architecture (JCA) implementation of an adapter. When an adapter is deployed to use the Connector for JCA, the Common Client Interface provides integration services using the adapter. For more information, see the *BEA WebLogic ERP Adapter Installation and Configuration* manual.

Procedure: How to Create iWay Business Services

To create iWay Business Services for an Oracle E-Business Suite table:

1. Connect to an Oracle E-Business Suite target, as described in *Opening a Connection to Oracle E-Business Suite* on page 2-3.

2. In the left pane, open the target node to display its modules.
The list includes all modules currently supported by Oracle E-Business Suite.
3. Open the module containing the table for which you want to create a business service.
4. Select the table.
The user ID for this target connection must have read access to the selected table, otherwise you cannot create a business service for it.
5. In the right pane, move the pointer over *Operations* and select *Create iWay Business Services*.

The Create Web Service pane opens on the right with the options to create a Web Service, as shown in the following image.



6. Select *Create a new service* and click *Next*.

The Create Web Service pane prompts you for a service name, description, and license, as shown in the following image.

Create Web Service for GL_INTERFACE

Service Name:

Description:

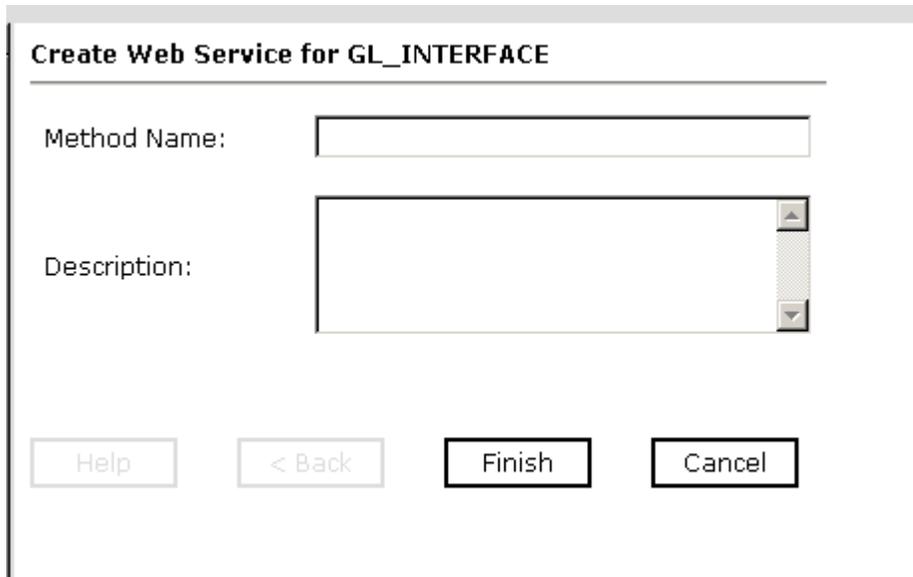
License:

- production
- test

Help < Back Next > Cancel

- a. In the Service Name field, type a descriptive name for the service.
 - b. In the Description field, type a brief description of the service.
 - c. From the License list, select a license definition.
7. Click Next.

The Create Web Services pane then prompts you for information about the method of the service, as shown in the following image.



Create Web Service for GL_INTERFACE

Method Name:

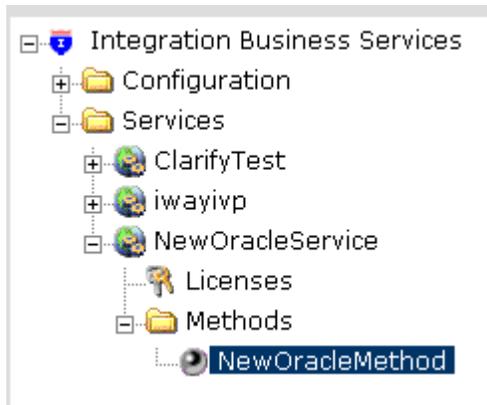
Description:

Help < Back Finish Cancel

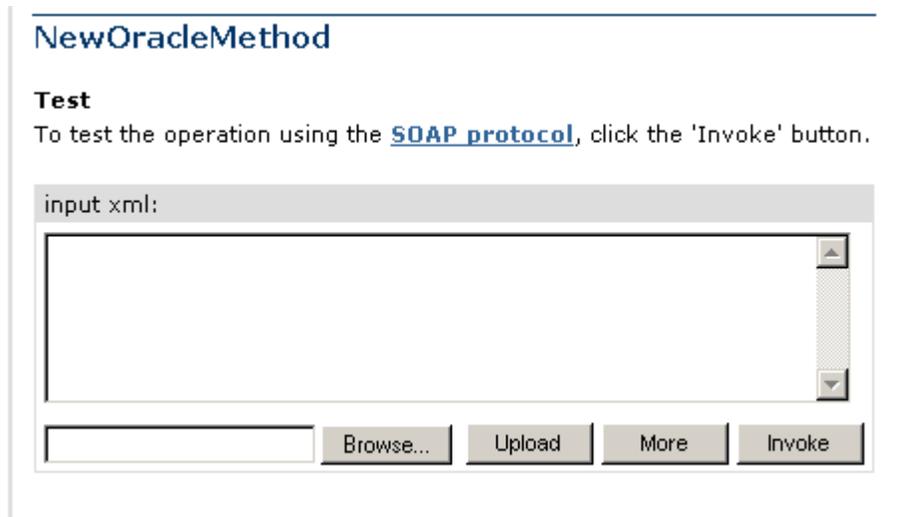
- a. In the Method Name field, type a descriptive name for the method.
 - b. In the Description field, type a brief description of the method.
8. Click *Finish*.

The iWay Business Services Engine tab opens. In the left pane, the new Web service appears under the *iWay Business Services, Services* node, with the new method appearing under the new Web service *Methods* node.

The following image shows a Web service example of NewOracleService and, under that service, a method example of NewOracleMethod.

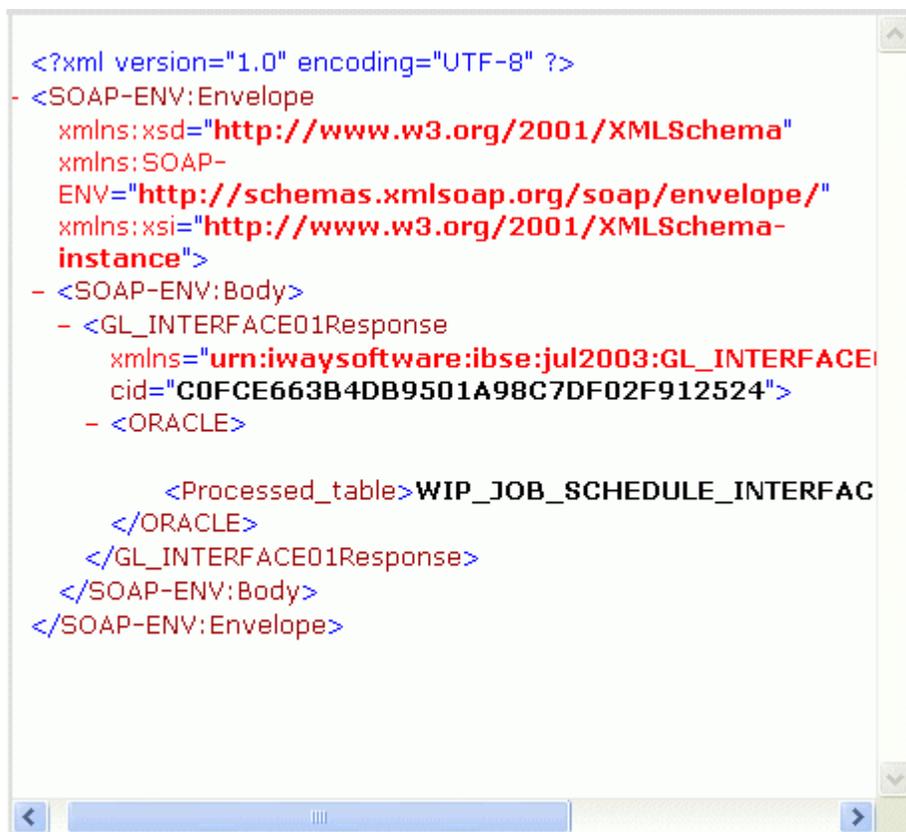


On the right, the test pane for the new method opens, as shown in the following image.



9. Enter a sample XML document that will query the service in the input xml field.
For sample input XML, see *Sample iWay Business Services Input XML* on page 2-33.
10. Click *Invoke*.

The Web service test result appears on the right pane. An example of a Web service test result is shown in the following image.



```
<?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance">
- <SOAP-ENV:Body>
  - <GL_INTERFACE01Response
    xmlns="urn:iwaysoftware:ibse:jul2003:GL_INTERFACE01"
    cid="C0FCE663B4DB9501A98C7DF02F912524">
    - <ORACLE>

      <Processed_table>WIP_JOB_SCHEDULE_INTERFAC
    </ORACLE>
    </GL_INTERFACE01Response>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example: Sample iWay Business Services Input XML

```

<ORACLE>
  <WIP_JOB_SCHEDULE_INTERFACE>
    <LAST_UPDATE_DATE>2002-12-02</LAST_UPDATE_DATE>
    <LAST_UPDATED_BY>1001611</LAST_UPDATED_BY>
    <CREATION_DATE>2002-12-02</CREATION_DATE>
    <CREATED_BY>1001611</CREATED_BY>
    <LAST_UPDATE_LOGIN/>
    <REQUEST_ID/>
    <PROGRAM_ID/>
    <PROGRAM_APPLICATION_ID/>
    <PROGRAM_UPDATE_DATE/>
    <GROUP_ID>2</GROUP_ID>
    <SOURCE_CODE/>
    <SOURCE_LINE_ID/>
    <PROCESS_TYPE/>
    <ORGANIZATION_ID/>
    <LOAD_TYPE>1</LOAD_TYPE>
    <STATUS_TYPE>3</STATUS_TYPE>
    <OLD_STATUS_TYPE/>
    <LAST_UNIT_COMPLETION_DATE/>
    <OLD_COMPLETION_DATE/>
    <PROCESSING_WORK_DAYS/>
    <DAILY_PRODUCTION_RATE/>
    <LINE_ID/>
    <PRIMARY_ITEM_ID>155</PRIMARY_ITEM_ID>
    <BOM_REFERENCE_ID/>
    <ROUTING_REFERENCE_ID/>
    <BOM_REVISION_DATE/>
    <ROUTING_REVISION_DATE/>
    <WIP_SUPPLY_TYPE>7</WIP_SUPPLY_TYPE>
    <CLASS_CODE>Discrete</CLASS_CODE>
    <LOT_NUMBER/>
    <LOT_CONTROL_CODE/>
    <JOB_NAME/>
    <DESCRIPTION/>
    <FIRM_PLANNED_FLAG/>
    <ALTERNATE_ROUTING_DESIGNATOR/>
    <ALTERNATE_BOM_DESIGNATOR> </ALTERNATE_BOM_DESIGNATOR>
    <START_QUANTITY>100</START_QUANTITY>
    <OLD_START_QUANTITY/>
    <WIP_ENTITY_ID/>
    <REPETITIVE_SCHEDULE_ID/>
    <ERROR/>
    <PARENT_GROUP_ID/>
    <ATTRIBUTE_CATEGORY/>
    <ATTRIBUTE1/>
    <ATTRIBUTE2/>

```

Generating a Business Service for an Interface Table

```
<ATTRIBUTE3 />
<ATTRIBUTE4 />
<ATTRIBUTE5 />
<ATTRIBUTE6 />
<ATTRIBUTE7 />
<ATTRIBUTE8 />
<ATTRIBUTE9 />
<ATTRIBUTE10 />
<ATTRIBUTE11 />
<ATTRIBUTE12 />
<ATTRIBUTE13 />
<ATTRIBUTE14 />
<ATTRIBUTE15 />
<INTERFACE_ID />
<LAST_UPDATED_BY_NAME />
<CREATED_BY_NAME />
<PROCESS_PHASE>2</PROCESS_PHASE>
<PROCESS_STATUS>1</PROCESS_STATUS>
<ORGANIZATION_CODE>M1</ORGANIZATION_CODE>
<FIRST_UNIT_START_DATE>2002-12-02</FIRST_UNIT_START_DATE>
<FIRST_UNIT_COMPLETION_DATE>2002-12-02</FIRST_UNIT_COMPLETION_DATE>
<LAST_UNIT_START_DATE>2002-12-02</LAST_UNIT_START_DATE>
<SCHEDULING_METHOD />
<LINE_CODE />
<PRIMARY_ITEM_SEGMENTS />
<BOM_REFERENCE_SEGMENTS />
<ROUTING_REFERENCE_SEGMENTS />
<ROUTING_REVISION />
<BOM_REVISION />
<COMPLETION_SUBINVENTORY />
<COMPLETION_LOCATOR_ID />
<COMPLETION_LOCATOR_SEGMENTS />
<SCHEDULE_GROUP_ID />
<SCHEDULE_GROUP_NAME />
<BUILD_SEQUENCE />
<PROJECT_ID />
<PROJECT_NAME />
<TASK_ID />
<TASK_NAME />
<NET_QUANTITY>100</NET_QUANTITY>
<DESCRIPTIVE_FLEX_SEGMENTS />
<PROJECT_NUMBER />
<TASK_NUMBER />
<PROJECT_COSTED />
<END_ITEM_UNIT_NUMBER />
<OVERCOMPLETION_TOLERANCE_TYPE />
<OVERCOMPLETION_TOLERANCE_VALUE />
<KANBAN_CARD_ID />
```

```

<PRIORITY>2</PRIORITY>
<DUE_DATE>2002-12-02</DUE_DATE>
<ALLOW_EXPLOSION>Y</ALLOW_EXPLOSION>
<HEADER_ID/>
<DELIVERY_ID/>
<COPRODUCTS_SUPPLY/>
<DUE_DATE_PENALTY/>
<DUE_DATE_TOLERANCE/>
</WIP_JOB_SCHEDULE_INTERFACE>
</ORACLE>

```

Procedure: How to Delete iWay Business Services

To delete iWay Business Services:

1. Select the iWay Business Services tab and, in the left pane, expand the *iWay Business Services Service* node.
2. Select the *Services* node.

Services information opens on the right, as shown in the following image. This pane lists and describes all available services.

Operations ▶

Services

The Integration Business Services Engine exposes functionality as web services. It acts as a gateway to heterogeneous back-end applications and databases.

Service name	Description
<input type="checkbox"/> ISO15022WS	This Web service transforms MT502 XML to MT500 format.
<input checked="" type="checkbox"/> iwayivp	This service is used to verify the installation of the Integration Business Services Engine. The methods provided by this service are predefined by the installation.
<input type="checkbox"/> QueryView	QuerywithView

3. Select the check box next to the service(s) you want to delete.

4. Move the pointer over *Operations* and select *Delete*.

A confirmation statement appears in the right pane.

5. Click *OK*.

The service no longer appears under the *Services* node on the left, or in the *Services* pane on the right.

Generating WSDL From a Web Service

Generating WSDL (Web Services Description Language) from a Web service enables you to make the Web service available to other services within a host server, such as BEA WebLogic Server.

Procedure: How to Generate WSDL From a Web Service

To generate WSDL from a Web service using Application Explorer:

1. Select the iWay Business Services tab.
2. In the left pane, expand the iWay Business Services node and the *Services* node.
3. Select the service for which you want to generate WSDL.

A link to a description of the service appears on the right.

4. Right-click the *Service Description* link and select *Save Target As*.

The Save As window opens.

5. Choose a location for the file and add a .wsdl file extension to the file name. All WSDL files must have a .wsdl file extension.
6. Click *Save*.

Saving a Web service creates a WSDL file, for example, saving a file named WIPMove, creates a WSDL file named WIPMove.wsdl.

Example: Generated WSDL File From a Web Service

The following is an example of a WSDL file:

```
<definitions xmlns:tns="urn:schemas-iwaysoftware-com:iwse"
targetNamespace="urn:schemas-iwaysoftware-com:iwse"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:m11="urn:iwaysoftware:ibse:jul2003:test:response"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:m1="urn:iwaysoftware:ibse:jul2003:test"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"><types><xs:schema
targetNamespace="urn:schemas-iwaysoftware-com:iwse"
elementFormDefault="qualified"><xs:element
name="ibsinfo"><xs:complexType><xs:sequence><xs:element type="xs:string"
name="service"/><xs:element type="xs:string" name="method"/><xs:element
type="xs:string" name="license"/><xs:element type="xs:string"
minOccurs="0" name="disposition"/><xs:element type="xs:string"
minOccurs="0" name="Username"/><xs:element type="xs:string" minOccurs="0"
name="Password"/><xs:element type="xs:string" minOccurs="0"
name="language"/></xs:sequence></xs:complexType>
</xs:element><xs:element
name="adapterexception"><xs:complexType><xs:sequence><xs:element
type="xs:string"
name="error"/></xs:sequence></xs:complexType></xs:element>
</xs:schema><xs:schema xml:lang="en"
targetNamespace="urn:iwaysoftware:ibse:jul2003:test"
attributeFormDefault="unqualified"
xmlns:m1="urn:iwaysoftware:ibse:jul2003:test"
elementFormDefault="qualified"><xs:element
name="test"><xs:complexType><xs:sequence><xs:element
name="ORACLE"><xs:complexType><xs:sequence><xs:element
name="WIP_JOB_SCHEDULE_INTERFACE"><xs:complexType><xs:sequence><xs:elemen
t minOccurs="1" name="LAST_UPDATE_DATE" maxOccurs="1"/><xs:element
minOccurs="0" name="LAST_UPDATED_BY" maxOccurs="1"/><xs:element
minOccurs="1" name="CREATION_DATE" maxOccurs="1"/><xs:element
minOccurs="0" name="CREATED_BY" maxOccurs="1"/><xs:element minOccurs="0"
name="LAST_UPDATE_LOGIN" maxOccurs="1"/><xs:element minOccurs="0"
name="REQUEST_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="PROGRAM_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="PROGRAM_APPLICATION_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="PROGRAM_UPDATE_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="GROUP_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="SOURCE_CODE" maxOccurs="1"/><xs:element minOccurs="0"
name="SOURCE_LINE_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="PROCESS_TYPE" maxOccurs="1"/><xs:element minOccurs="0"
name="ORGANIZATION_ID" maxOccurs="1"/><xs:element minOccurs="1"
```

Generating a Business Service for an Interface Table

```
name="LOAD_TYPE" maxOccurs="1"/><xs:element minOccurs="0"
name="STATUS_TYPE" maxOccurs="1"/><xs:element minOccurs="0"
name="OLD_STATUS_TYPE" maxOccurs="1"/><xs:element minOccurs="0"
name="LAST_UNIT_COMPLETION_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="OLD_COMPLETION_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="PROCESSING_WORK_DAYS" maxOccurs="1"/><xs:element minOccurs="0"
name="DAILY_PRODUCTION_RATE" maxOccurs="1"/><xs:element minOccurs="0"
name="LINE_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="PRIMARY_ITEM_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="BOM_REFERENCE_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="ROUTING_REFERENCE_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="BOM_REVISION_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="ROUTING_REVISION_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="WIP_SUPPLY_TYPE" maxOccurs="1"/><xs:element minOccurs="0"
name="CLASS_CODE" maxOccurs="1"/><xs:element minOccurs="0"
name="LOT_NUMBER" maxOccurs="1"/><xs:element minOccurs="0"
name="LOT_CONTROL_CODE" maxOccurs="1"/><xs:element minOccurs="0"
name="JOB_NAME" maxOccurs="1"/><xs:element minOccurs="0"
name="DESCRIPTION" maxOccurs="1"/><xs:element minOccurs="0"
name="FIRM_PLANNED_FLAG" maxOccurs="1"/><xs:element minOccurs="0"
name="ALTERNATE_ROUTING_DESIGNATOR" maxOccurs="1"/><xs:element
minOccurs="0" name="ALTERNATE_BOM_DESIGNATOR" maxOccurs="1"/><xs:element
minOccurs="0" name="DEMAND_CLASS" maxOccurs="1"/><xs:element
minOccurs="0" name="START_QUANTITY" maxOccurs="1"/><xs:element
minOccurs="0" name="OLD_START_QUANTITY" maxOccurs="1"/><xs:element
minOccurs="0" name="WIP_ENTITY_ID" maxOccurs="1"/><xs:element
minOccurs="0" name="REPETITIVE_SCHEDULE_ID" maxOccurs="1"/><xs:element
minOccurs="0" name="ERROR" maxOccurs="1"/><xs:element minOccurs="0"
name="PARENT_GROUP_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE_CATEGORY" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE1" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE2" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE3" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE4" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE5" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE6" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE7" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE8" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE9" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE10" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE11" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE12" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE13" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE14" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE15" maxOccurs="1"/><xs:element minOccurs="0"
name="INTERFACE_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="LAST_UPDATED_BY_NAME" maxOccurs="1"/><xs:element minOccurs="0"
name="CREATED_BY_NAME" maxOccurs="1"/><xs:element minOccurs="0"
```

```

name="PROCESS_PHASE" maxOccurs="1"/><xs:element minOccurs="0"
name="PROCESS_STATUS" maxOccurs="1"/><xs:element minOccurs="0"
name="ORGANIZATION_CODE" maxOccurs="1"/><xs:element minOccurs="0"
name="FIRST_UNIT_START_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="FIRST_UNIT_COMPLETION_DATE" maxOccurs="1"/><xs:element
minOccurs="0" name="LAST_UNIT_START_DATE" maxOccurs="1"/><xs:element
minOccurs="0" name="SCHEDULING_METHOD" maxOccurs="1"/><xs:element
minOccurs="0" name="LINE_CODE" maxOccurs="1"/><xs:element minOccurs="0"
name="PRIMARY_ITEM_SEGMENTS" maxOccurs="1"/><xs:element minOccurs="0"
name="BOM_REFERENCE_SEGMENTS" maxOccurs="1"/><xs:element minOccurs="0"
name="ROUTING_REFERENCE_SEGMENTS" maxOccurs="1"/><xs:element
minOccurs="0" name="ROUTING_REVISION" maxOccurs="1"/><xs:element
minOccurs="0" name="BOM_REVISION" maxOccurs="1"/><xs:element
minOccurs="0" name="COMPLETION_SUBINVENTORY" maxOccurs="1"/><xs:element
minOccurs="0" name="COMPLETION_LOCATOR_ID" maxOccurs="1"/><xs:element
minOccurs="0" name="COMPLETION_LOCATOR_SEGMENTS"
maxOccurs="1"/><xs:element minOccurs="0" name="SCHEDULE_GROUP_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="SCHEDULE_GROUP_NAME"
maxOccurs="1"/><xs:element minOccurs="0" name="BUILD_SEQUENCE"
maxOccurs="1"/><xs:element minOccurs="0" name="PROJECT_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="PROJECT_NAME"
maxOccurs="1"/><xs:element minOccurs="0" name="TASK_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="TASK_NAME"
maxOccurs="1"/><xs:element minOccurs="0" name="NET_QUANTITY"
maxOccurs="1"/><xs:element minOccurs="0" name="DESCRIPTIVE_FLEX_SEGMENTS"
maxOccurs="1"/><xs:element minOccurs="0" name="PROJECT_NUMBER"
maxOccurs="1"/><xs:element minOccurs="0" name="TASK_NUMBER"
maxOccurs="1"/><xs:element minOccurs="0" name="PROJECT_COSTED"
maxOccurs="1"/><xs:element minOccurs="0" name="END_ITEM_UNIT_NUMBER"
maxOccurs="1"/><xs:element minOccurs="0"
name="OVERCOMPLETION_TOLERANCE_TYPE" maxOccurs="1"/><xs:element
minOccurs="0" name="OVERCOMPLETION_TOLERANCE_VALUE"
maxOccurs="1"/><xs:element minOccurs="0" name="KANBAN_CARD_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="PRIORITY"
maxOccurs="1"/><xs:element minOccurs="0" name="DUE_DATE"
maxOccurs="1"/><xs:element minOccurs="0" name="ALLOW_EXPLOSION"
maxOccurs="1"/><xs:element minOccurs="0" name="HEADER_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="DELIVERY_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="COPRODUCTS_SUPPLY"
maxOccurs="1"/><xs:element minOccurs="0" name="DUE_DATE_PENALTY"
maxOccurs="1"/><xs:element minOccurs="0" name="DUE_DATE_TOLERANCE"
maxOccurs="1"/><xs:element minOccurs="0" name="XML_DOCUMENT_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="PARENT_WIP_ENTITY_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="PARENT_JOB_NAME"
maxOccurs="1"/><xs:element minOccurs="0" name="ASSET_NUMBER"
maxOccurs="1"/><xs:element minOccurs="0" name="ASSET_GROUP_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="ASSET_GROUP_SEGMENTS"
maxOccurs="1"/><xs:element minOccurs="0" name="PM_SCHEDULE_ID"

```

Generating a Business Service for an Interface Table

```
maxOccurs="1"/><xs:element minOccurs="0" name="REBUILD_ITEM_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="REBUILD_ITEM_SEGMENTS"
maxOccurs="1"/><xs:element minOccurs="0" name="REBUILD_SERIAL_NUMBER"
maxOccurs="1"/><xs:element minOccurs="0" name="MANUAL_REBUILD_FLAG"
maxOccurs="1"/><xs:element minOccurs="0" name="SHUTDOWN_TYPE"
maxOccurs="1"/><xs:element minOccurs="0" name="NOTIFICATION_REQUIRED"
maxOccurs="1"/><xs:element minOccurs="0" name="WORK_ORDER_TYPE"
maxOccurs="1"/><xs:element minOccurs="0" name="OWNING_DEPARTMENT"
maxOccurs="1"/><xs:element minOccurs="0" name="OWNING_DEPARTMENT_CODE"
maxOccurs="1"/><xs:element minOccurs="0" name="ACTIVITY_TYPE"
maxOccurs="1"/><xs:element minOccurs="0" name="ACTIVITY_CAUSE"
maxOccurs="1"/><xs:element minOccurs="0" name="TAGOUT_REQUIRED"
maxOccurs="1"/><xs:element minOccurs="0" name="PLAN_MAINTENANCE"
maxOccurs="1"/><xs:element minOccurs="0" name="DATE_RELEASED"
maxOccurs="1"/><xs:element minOccurs="0" name="REQUESTED_START_DATE"
maxOccurs="1"/></xs:sequence></xs:complexType></xs:element></xs:sequence>
</xs:complexType></xs:element></xs:sequence></xs:complexType></xs:element
>

</xs:schema><xs:schema xml:lang="en"
targetNamespace="urn:iwaysoftware:ibse:jul2003:test:response"
attributeFormDefault="unqualified"
xmlns:m11="urn:iwaysoftware:ibse:jul2003:test:response"
elementFormDefault="qualified"><xs:element
name="testResponse"><xs:complexType><xs:sequence><xs:element
name="ORACLE"><xs:complexType><xs:sequence><xs:element minOccurs="0"
name="Processed_table" maxOccurs="unbounded"/><xs:element minOccurs="0"
name="submit_response_id" maxOccurs="unbounded"/><xs:element
name="Return_Code" /></xs:sequence></xs:complexType></xs:element>

</xs:sequence><xs:attribute type="xs:string"
use="required" name="cid" /></xs:complexType></xs:element></xs:schema>

</types><message name="testIn"><part element="m1:test"
name="parameters" />

</message><message name="testOut"><part element="m11:testResponse"
name="parameters" />

</message><message name="testserviceHeader"><part
element="tns:ibsinfo" name="header" />

</message><message name="AdapterException"><part
element="tns:adapterexception" name="fault" />

</message><portType name="testserviceSoap"><operation
name="test"><documentation/><input message="tns:testIn" /><output
message="tns:testOut" /><fault message="tns:AdapterException"
name="AdapterExceptionFault" /></operation>
```

```

    </portType><binding type="tns:testserviceSoap"
name="testserviceSoap"><soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/><operation
name="test"><soap:operation style="document"
soapAction="testservice.testRequest@test@"/><input><soap:body
use="literal"/><soap:header part="header" message="tns:testserviceHeader"
use="literal"/>

    </input><output><soap:body use="literal"/>

    </output><fault name="AdapterExceptionFault"><soap:fault
use="literal" name="AdapterExceptionFault"/></fault></operation>

    </binding><service
name="testservice"><documentation>testservice</documentation><port
binding="tns:testserviceSoap" name="testserviceSoap1"><soap:address
location="http://111KLEINMAN.ibi.com:7001/ibse/IBSEServlet/XDSOAPRouter"/
></port></service></definitions>

```

Identity Propagation

If you test or execute a Web service using a third party XML editor, for example XMLSPY, the Username and Password values that you specify in the SOAP header must be valid and are used to connect to Oracle. The user name and password values that you provided for Oracle during target creation using Application Explorer are overwritten for this Web service request. The following is a sample SOAP header that is included in the WSDL file for a Web service:

```

<SOAP-ENV:Header>
  <m:ibsinfo xmlns:m="urn:schemas-iwaysoftware-com:iwse">
    <m:service>String</m:service>
    <m:method>String</m:method>
    <m:license>String</m:license>
    <m:disposition>String</m:disposition>
    <m:Username>String</m:Username>
    <m>Password>String</m>Password>
    <m:language>String</m:language>
  </m:ibsinfo>
</SOAP-ENV:Header>

```

Note: You can remove the following tags from the SOAP header, since they are not required:

```

<m:disposition>String</m:disposition>

<m:language>String</m:language>

```

Running a Submit Request

The adapter supports running what was formerly known as the Oracle Applications concurrent program function using the adapter concurrent programs agent. Currently, this function is called the Submit Request.

The Submit Request submits a request that runs a concurrent program. Concurrent programs perform several functions, including moving data. For example, if you wish to move data to a base table:

1. First, write the data to an interface table.

Each interface table has a Submit Request associated with it.

2. When the data is ready, run a Submit Request.

The Submit Request moves the data from the interface table to the base table.

You can run a Submit Request:

- Immediately, using the Submit Request or using the Oracle E-Business Suite Web Client.
- At specifically scheduled times, using the Oracle E-Business Suite Web Client.

The Submit Request can call a concurrent program synchronously or asynchronously. It submits the concurrent program to a concurrent manager that controls background processing in Oracle E-Business Suite.

Procedure: How to Run the Submit Request

To run the Submit Request:

1. Use Application Explorer to generate a pair of request and response schemas for the Submit Request, as described in *Generating Schema for the Submit Request on page 2-44*.

You do this only once for a Submit Request. This covers all Submit Requests to be run from a given Oracle E-Business Suite connection.

2. Add a service for the Submit Request, as described in *Adding a Service for the Submit Request on page 2-44*.

You do this only once for a Submit Request. This covers all Submit Requests to be run from one Oracle E-Business Suite connection.

3. Create the request document, as described in *Creating a Request Document on page 2-49*.

4. Test the request document, as described in *Testing the Configuration for the Submit Request on page 2-50*.

Installing the Submit Request and Running the Concurrent.ora Script

To install the Submit Request, you must perform the following additional steps when installing the adapter:

1. Run the Concurrent.ora script against the Oracle database server.
2. Adjust the system path on the Oracle database server.
3. Verify the presence of the CONCSUB executable on the Oracle database server.

As part of the installation process, you must run the Concurrent.ora script. You can run it at any time before creating the service for the Submit Request. The computer on which you run the script must have:

- A TNSnames entry, in the Tnsnames.ora file, that points to the Oracle E-Business Suite database instance (on the Oracle server).
- SQL*Plus installed.

Procedure: How to Run the Concurrent.ora Script

To run the Concurrent.ora script:

1. On Windows, use WinZip (or a similar extraction product) to extract Concurrent.ora from the adapter's EAR file.
2. After Concurrent.ora is extracted, move it to the computer where SQL*Plus resides, if it is not already located there.
3. Open a command prompt window or telnet session.
4. In the directory in which Concurrent.ora resides, issue the following command:

```
SQLPLUS APPS/password@database @Concurrent.ora
```

where:

password

Is the password associated with the APPS user ID.

database

Is a value of the tnsnames entry (in the tnsnames.ora file) that points to the Oracle E-Business Suite database instance.

Ensure you issue the command using the APPS user ID.

Procedure: How to Adjust the System Path and Verify CONCSUB

To adjust the system path and verify CONCSUB:

1. On the server where the Oracle E-Business Suite database instance resides, add the following to the system path:

```
$FND_TOP\bin
```

You can add it at any time before creating the service for the Submit Request.

2. Verify that the CONCSUB executable is present in \$FND_TOP\bin.

If CONCSUB is missing, contact your Oracle E-Business Suite administrator.

\$FND_TOP is the location of the Oracle E-Business Suite foundation directory. Its value is set in the vis.env (or prod.env) file located under the oraclehome\visappl directory.

Generating Schema for the Submit Request

You must create a pair of request and response service schemas for the Submit Request. You need only do this once for each Oracle E-Business Suite connection from which you run these programs. This satisfies the requirement for all Submit Requests that are accessed using that connection.

Procedure: How to Generate Schema for the Submit Request

To generate the request and response schemas:

1. If you have not already done so, connect to an Oracle E-Business Suite database instance.
2. In the left pane of Application Explorer under the Oracle Applications target, expand the Services node and the *submit_request* node.
3. Select *submit_request*.
4. In the right pane, move the pointer over *Operations* and select *Generate Schema*.
A table appears on the right.
5. To review a schema, click the ellipsis (...) in the Schemas column for the appropriate row.

You are finished generating the service schemas.

Adding a Service for the Submit Request

Adding a service for the Submit Request is identical to configuring other Oracle E-Business Suite services, except that some of the configuration parameters differ.

The following table lists and describes the service parameters for the Submit Request.

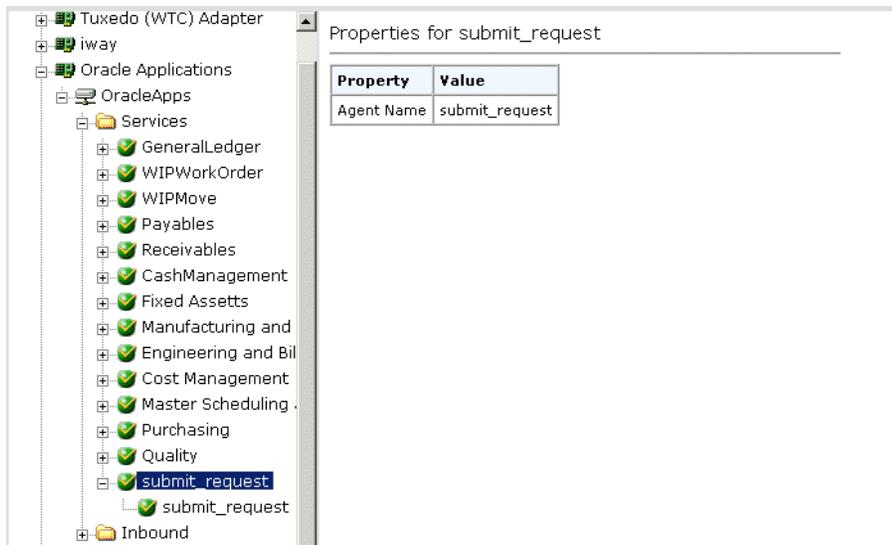
Parameter	Description
Select	Submit Request.
Userid	User ID for the Oracle E-Business Suite Suite database.
Password	Password associated with the user ID.
Host	Name of the server where the Oracle E-Business Suite database instance resides.
Port	Port number on which the database is listening.
ServiceId	Unique name, comprised of the database name and domain name, that identifies the database.
TnsName	Logical name of the Oracle E-Business Suite database instance.

Procedure: How to Add a Submit Request Service

To add a Submit Request service:

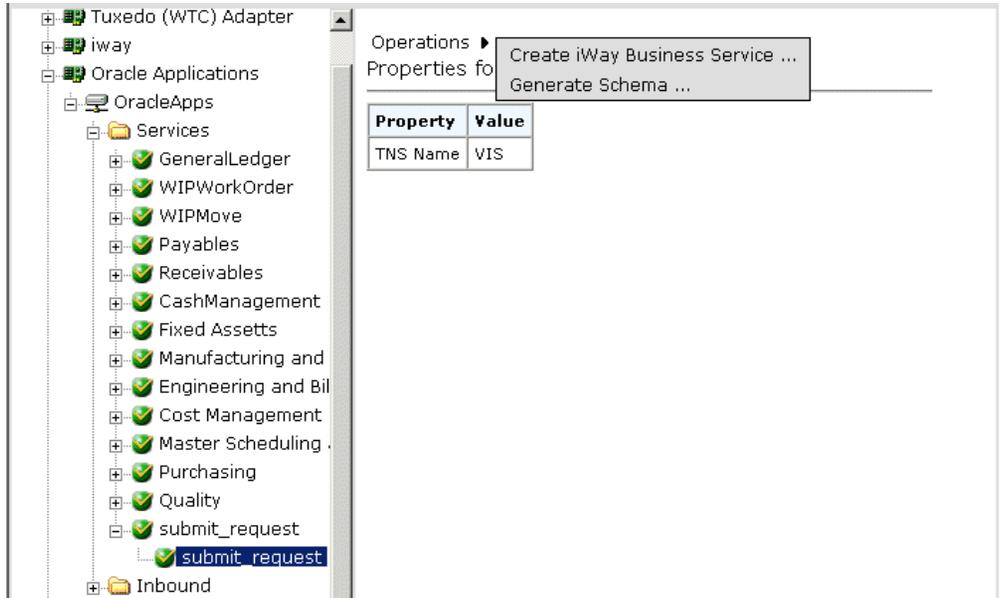
1. In the left pane of Application Explorer, expand the *submit_request* node.

The following image shows the expanded *submit_request* service node on the left and a property table for the submit request service on the right.



2. Select *submit_request*.

The Operations menu becomes available on the right, as shown in the following image.



3. Move the pointer over *Operations* and select *Create iWay Business Services*.

The Create Web Service for *submit_request* pane opens on the right. This pane, shown in the following image, provides the option to create a new service or use an existing service.



4. Select the *Create a new service* option button.
5. Click *Next*.

The right pane changes to prompt you for descriptive information about the service, as shown in the following image.

Create Web Service for submit_request

Service Name:

Description:

License:

- a. In the Name field, type a name for the Web service.
- b. In the Description field, type a brief description of the Web service.
- c. From the License list, select the license, for example, *test*.

6. Click *Next*.

The right pane changes again to prompt you to identify a method for the service, as shown in the following image.

The image shows a dialog box titled "Create Web Service for submit_request". It contains two input fields: "Method Name" with the text "submit_request" and "Description" with the text "execute concurrent program". Below the fields are four buttons: "Help", "< Back", "Finish", and "Cancel".

- a.** In the Method Name field, type a name for your method.
- b.** In the Description field, type a brief description of the method.

7. Click *Finish*.

Creating a Request Document

Use an XML editor to create the service request document. The XML request document for a Submit Request has the following syntax

```
<Oracle>
  <submit_request>
    <resp_appl_shrtnm/>
    <responsibility/>
    <username/>
    <wait/>
    <prog_appl_shrtnm/>
    <program/>
    <parm/>
    <parm/>
    <parm/>
    . . .
    <parm/>
  </submit_request>
</Oracle>
```

where:

resp_appl_shrtnm

Is the application short name of your responsibility. SYSADMIN is the default value.

responsibility

Is the responsibility of the application user. System Administrator is the default value.

username

Is the name of your Application Object Library user. This name is used to update WHO information for data that the Submit Request changes and to create the report output file name for the specific request. This parameter is different from the user ID that is used to log on to the Oracle database. SYSADMIN is the default value.

wait

Is the number of seconds to wait between checks of the status of the Submit Request. This must be a positive integer, or Y or N. The value, Y, indicates that status is checked every 60 seconds. N is the default value.

prog_appl_shrtnm

Is the application short name of your program. This value cannot be null.

program

Is the name of the Submit Request. This value cannot be null.

parm

Is a concurrent program parameter.

The following table, which lists the parameters with possible values, can assist you in determining parameter values. This information changes from release to release, and so may differ for a particular installation. Use it as a guide in obtaining installation application parameter values, not as a reference.

Parameter	Possible Value
responsibility	RESPONSIBILITY_ID in FND_RESPONSIBILITY and FND_RESPONSIBILITY_TL
username	USER_ID in FND_USER
resp_appl_shrtnm	APPLICATION_SHORT_NAME in FND_APPLICATION
program	CONCURRENT_PROGRAM_NAME in FND_CONCURRENT_PROGRAMS and FND_CONCURRENT_PROGRAMS_T

For example, a request for running the WIP Mass Load concurrent program looks similar to the following:

```
<ORACLE>
  <submit_request>
    <resp_appl_shrtnm>SYSADMIN</resp_appl_shrtnm>
    <responsibility>System Administrator</responsibility>
    <username>SYSADMIN</username>
    <wait>N</wait>
    <prog_appl_shrtnm>WICMPL</prog_appl_shrtnm>
    <program>WICMLX</program>
    <argument1>3993</argument1>
    <argument2>0</argument2>
    <argument3>1</argument3>
  </submit_request>
</ORACLE>
```

Testing the Configuration for the Submit Request

After you create a service and the request document, you can verify that you correctly configured the service for the Submit Request.

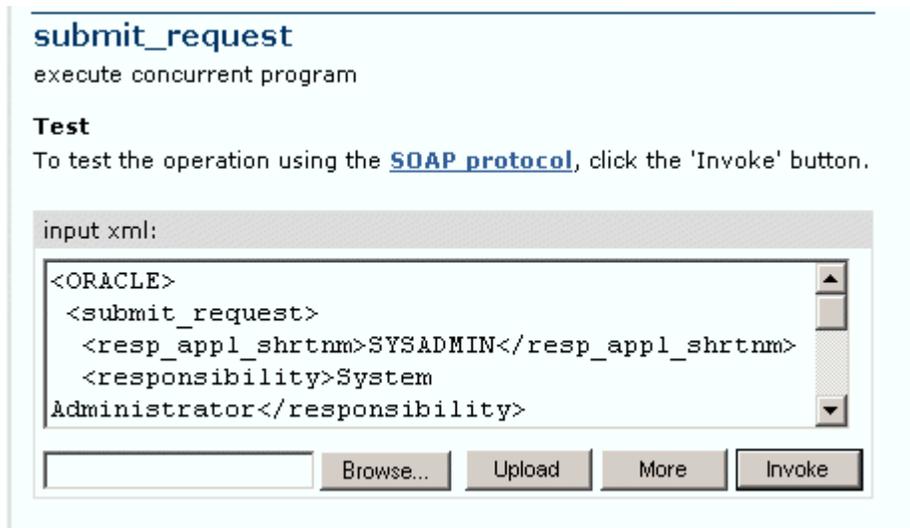
Procedure: How to Test the Submit Request

To test the Submit Request service:

1. When you create a service, the test pane appears on the right.
If the Test pane does not appear after you create the service:
 - a. Click the *iWay Business Services* tab.

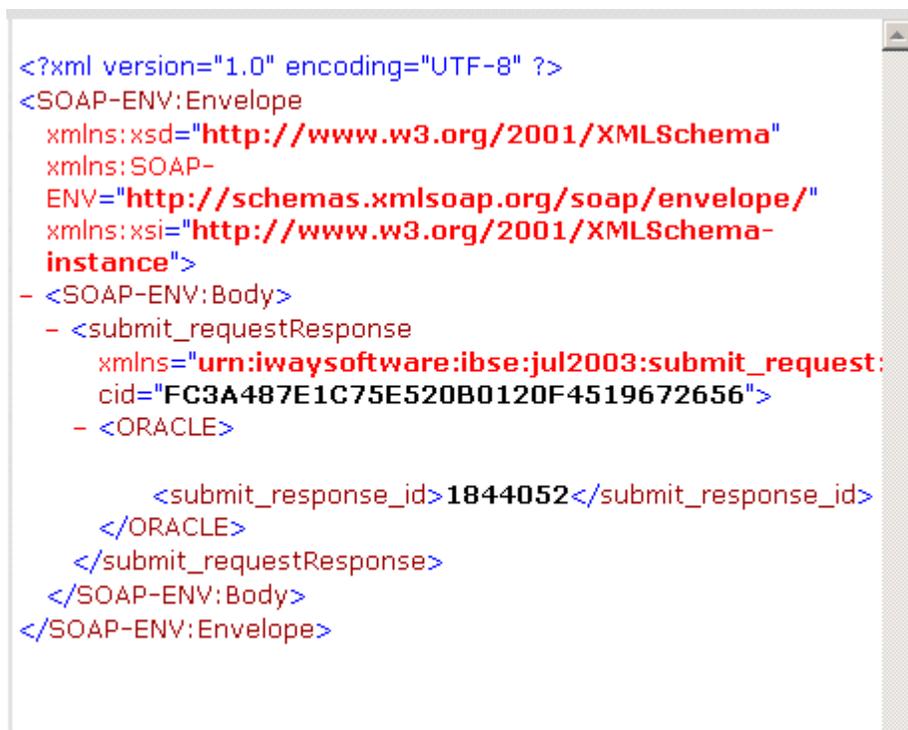
- b. Click the *Browse* button.
 - c. Use the tree in the left pane to browse to your service and method.
2. Copy the XML input from the request document to the input xml field of the test pane.

The following image shows the test pane for `submit_request` with an example of XML input.



3. To test the Submit Request, click *Invoke*.

The response document appears on the right. An example of a response document for the service is shown in the following image.

A screenshot of a text editor window showing a SOAP response document. The document is XML-formatted with various namespaces and elements. The root element is <?xml version="1.0" encoding="UTF-8" ?>. The envelope is <SOAP-ENV:Envelope> with namespaces for xsd, SOAP-ENV, and xsi. The body contains a <submit_requestResponse> element with an ORACLE sub-element containing a <submit_response_id>1844052</submit_response_id> element.

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance">
- <SOAP-ENV:Body>
- <submit_requestResponse
  xmlns="urn:iwaysoftware:ibse:jul2003:submit_request:
  cid="FC3A487E1C75E520B0120F4519672656">
- <ORACLE>

      <submit_response_id>1844052</submit_response_id>
    </ORACLE>
  </submit_requestResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Testing of the Submit Request service is complete.

CHAPTER 3

Listening for Oracle E-Business Suite Events

Topics:

- Understanding Event Functionality
- Creating an Event Port
- Creating a Channel
- Choosing a Listening Technique

Servlet Application Explorer enables you to listen for events in Oracle E-Business Suite interface tables. Several listening techniques are available, enabling you to choose the technique that best suits your requirements.

Although this section describes the Java servlet implementation of Application Explorer, other implementations provide the same functionality by means of similar graphical user interfaces. For information on using Application Explorer in BEA WebLogic Workshop, see Appendix B, *Using Application Explorer in BEA WebLogic Workshop for Event Handling*.

Understanding Event Functionality

Events are generated as a result of Oracle E-Business Suite activity. You can use events to trigger an action in your application. For example, you can detect WIP discrete job creation and notify another enterprise integration system of the event.

After you create a connection to your application system, you can add events using Servlet Application Explorer. To create an event, you must create a port and a channel.

- **Port**

A port associates a particular business object exposed by an adapter with a particular disposition. A disposition defines the protocol and location of the event data. The port defines the end point of the event consumption.

For example, you can use a JMS protocol to route the result of polling an interface table to a JMS queue hosted by BEA WebLogic Server. For more information, see *Creating an Event Port* on page 3-2.

- **Channel**

A channel represents configured connections to particular instances of back-end or other types of systems. A channel binds one or more event ports to a particular listener managed by an adapter. For more information, see *Creating a Channel* on page 3-21.

You can employ several techniques when listening for Oracle E-Business Suite events, depending upon your requirements. For information about these techniques, see *Choosing a Listening Technique* on page 3-29.

Creating an Event Port

The following procedures describe how to create an event port using Servlet Application Explorer.

When you use Application Explorer with an iWay Business Services Engine (iBSE) implementation, the following port dispositions are available:

- File
- iBSE
- MSMQ
- JMSQ
- SOAP
- HTTP
- MQSeries

When you use Application Explorer with a JCA implementation, the following port dispositions are available:

- File
- JMS
- MQSeries
- HTTP

Procedure: How to Create an Event Port for File

To create an event port for File:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node and the *Oracle Applications* node.
3. Select the *ports* node.

The following image shows the ports node selected on the left and the Operations menu open on the right.



4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane, shown in the following image, opens on the right providing four fields to identify the new port.

Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- a. In the Port Name field, type a name for the event port.
- b. In the Description field, type a brief description of the port.
- c. From the Disposition Protocol drop-down list, select *FILE*.
- d. In the Disposition field, specify a destination file to which the event data is written.

When pointing Application Explorer to an **IBSE** deployment, specify the destination file using the following format:

```
ifile://location[;errorTo=errorDest]
```

When pointing Application Explorer to a **JCA** deployment, specify the destination file using the following format:

```
location
```

The following table lists and describes the disposition parameters for FILE.

Parameter	Description
<i>location</i>	Full directory path and file name to which the data is written.
<i>errorDest</i>	Location to which error logs are sent. Optional. This can be a predefined port name or another disposition URL. The URL must be complete, including the protocol.

For example:

```
ifile://c:\temp\OraEvent.txt;errorTo=ifile://c:\temp\error
```

5. Click OK.

In the left pane, the event port appears under the ports node. In the right pane, summary information associated with the event port you created appears. An example of a port listing and summary information is shown in the following image.



You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-21.

Procedure: How to Create an Event Port for iBSE

To create an event port for iBSE:

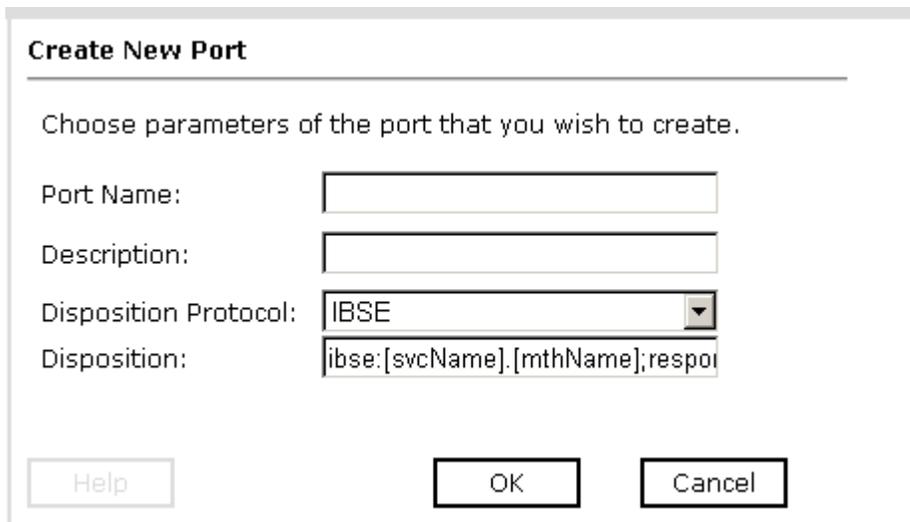
1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node and the *Oracle Applications* node.
3. Select the *ports* node.

The following image shows the ports node selected on the left and the Operations menu open on the right



4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane, shown in the following image, opens on the right providing four fields to identify the new port.



- a. In the Port Name field, type a name for the event port.
- b. In the Description field, type a brief description of the port.
- c. From the Disposition Protocol drop-down list, select *IBSE*.
- d. In the Disposition field, enter an iBSE destination using the following format:

`ibse: / svcName.mthName[; responseTo=respDest] [; errorTo=errorDest]`

The following table lists and describes the disposition parameters for IBSE.

Parameter	Description
<i>svcName</i>	Name of the service created with iBSE.
<i>methName</i>	Name of the method created for the Web service.
<i>respDest</i>	Location to which responses to the Web service are posted. Optional. This can be a predefined port name or another disposition URL. The URL must be complete, including the protocol.
<i>errorDest</i>	Location to which error logs are sent. Optional. This can be a predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click OK.

In the left pane, the event port appears under the ports node. In the right pane, summary information associated with the event port you created appears. An example of a port listing and summary information is shown in the following image.



You are now ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-21.

Procedure: How to Create an Event Port for MSMQ

To create an event port for a Microsoft Message Queuing (MSMQ) queue using Application Explorer:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node and the *Oracle Applications* node.

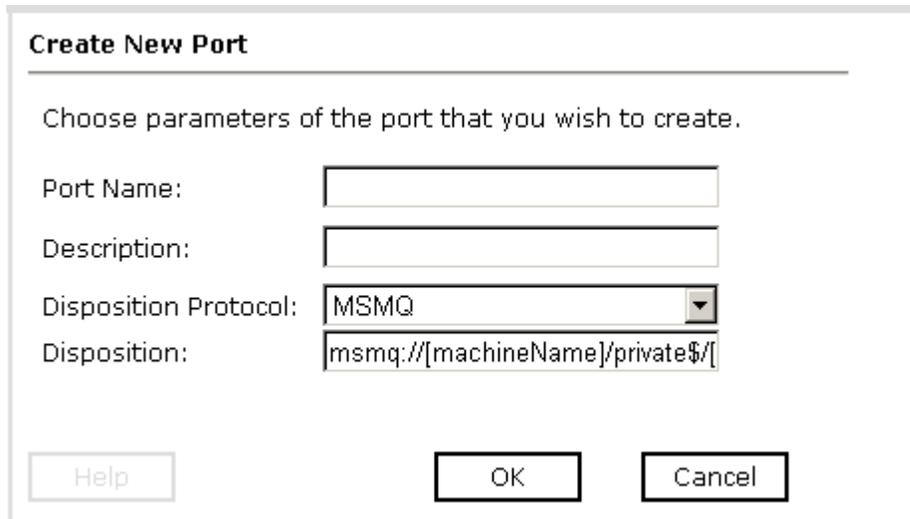
3. Select the *ports* node.

The following image shows the ports node selected on the left and the Operations menu open on the right



4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane, shown in the following image, opens on the right providing four fields to identify the new port.



- a. In the Port Name field, type a name for the event port.
- b. In the Description field, type a description of the port.
- c. From the Disposition Protocol drop-down list, select *MSMQ*.
- d. In the Disposition field, enter an MSMQ destination using the following format:

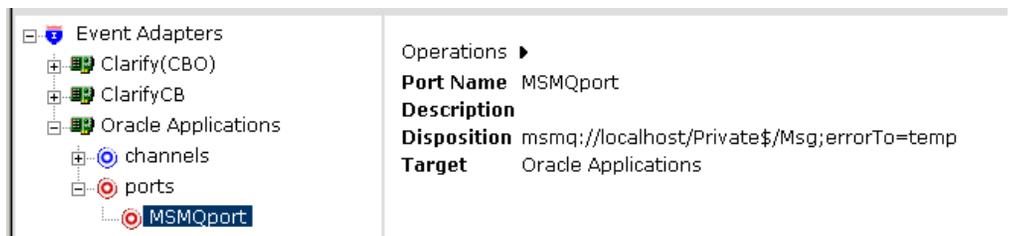
`msmq: /host/queueType/queueName[;errorTo=errorDest]`

The following table lists and describes the disposition parameters for MSMQ.

Parameter	Description
<i>host</i>	Name of the host on which the Microsoft Message Queuing system runs.
<i>queueType</i>	Type of queue. For private queues, enter <i>Private\$</i> . Private queues are queues that are not published in Active Directory. They appear only on the local computer that contains them. Private queues are accessible only by Message Queuing applications that recognize the full path name or format name of the queue.
<i>queueName</i>	Name of the queue in which messages are placed.
<i>errorDest</i>	Location to which error logs are sent. Optional. This can be a predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click **OK**.

In the left pane, the event port appears under the ports node. In the right pane, summary information associated with the event port you created appears. An example of a port listing and summary information is shown in the following image.



You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-21.

Procedure: How to Create an Event Port for JMSQ

To create an event port for a JMS queue using Application Explorer:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node and the *Oracle Applications* node.

3. Select the *ports* node.

The following image shows the ports node selected on the left and the Operations menu open on the right



4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane, shown in the following image, opens on the right providing four fields to identify the new port.

A screenshot of a dialog box titled 'Create New Port'. The dialog box has a title bar and a main area with the following text: 'Choose parameters of the port that you wish to create.' Below this text are four input fields: 'Port Name:', 'Description:', 'Disposition Protocol:', and 'Disposition:'. The 'Disposition Protocol:' field is a drop-down menu with 'JMSQ' selected. The 'Disposition:' field contains the text 'jmsq:[myQueueName]@[myQueueName]'. At the bottom of the dialog box are three buttons: 'Help', 'OK', and 'Cancel'.

- a. In the Port Name field, type a name for the event port.
- b. In the Description field, type a brief description of the port.
- c. In the Disposition Protocol drop-down list, select *JMSQ*.
- d. In the Disposition field, enter a JMS destination.

When pointing Application Explorer to an **ibSE** deployment, specify the destination file using the following format:

```
jmsq:queue@conn_factory;jndiurl=jndi_url;jndifactory=jndi_factory;
user=userID;password=pass[;errorTo=errorDest]
```

When pointing Application Explorer to a **JCA** deployment, specify the destination file using the following format:

```
jms:queue@conn_factory;jndiurl=jndi_url;jndifactory=jndi_factory
```

The following table lists and describes the disposition parameters for JMS.

Parameter	Description
<i>queue</i>	Name of a queue to which events are emitted.
<i>conn_factory</i>	Connection factory, a resource that contains information about the JMS Server. The BEA WebLogic connection factory is: <code>javax.jms.QueueConnectionFactory</code>
<i>jndi_url</i>	The URL to use to contact the JNDI provider. The syntax of this URL depends on which JNDI provider is used. This value corresponds to the standard JNDI property, <code>java.naming.provider.url</code> For BEA WebLogic Server, this is <code>t3://host:port</code> where: <i>host</i> Is the machine name where BEA WebLogic Server is installed. <i>port</i> Is the port on which BEA WebLogic Server is listening. The default port if not changed at installation is 7001.
<i>jndi_factory</i>	Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider. For BEA WebLogic Server, the WebLogic factory is <code>weblogic.jndi.WLInitialContextFactory</code>
<i>userID</i>	User ID associated with this queue.

Parameter	Description
<i>pass</i>	Password associated with the user ID.
<i>errorDest</i>	Location to which error logs are sent. Optional. This can be a predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

In the left pane, the event port appears under the ports node. In the right pane, summary information associated with the event port you created appears. An example of a port listing and summary information is shown in the following image.



Use the scroll bar at the bottom of the Operations pane to see the entire Disposition data.

You are now ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-21.

Procedure: How to Create an Event Port for SOAP

To create a port for a SOAP disposition:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node and the *Oracle Applications* node.
3. Select the *ports* node.

The following image shows the ports node selected on the left and the Operations menu open on the right



4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane, shown in the following image, opens on the right providing four fields to identify the new port.

Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- a. In the Port Name field, type a name for the event port.
- b. In the Description field, type a brief description of the port.
- c. From the Disposition Protocol drop-down list, select *SOAP*.
- d. In the Disposition field, enter a SOAP destination using the following format:

```
soap:[wsdl-url];soapaction=[myaction];method=[web service
method];namespace=[namespace];responseTo=[pre-defined port name or
another disposition URL];errorTo=[pre-defined port name or another
disposition url]
```

The following table defines the disposition parameters for SOAP.

Parameter	Description
<code>wSDL-url</code>	<p>The URL to the WSDL file that is required to create the SOAP message. For example:</p> <p><code>http://localhost:7001/ibse/IBSEServlet/test/webservice.ibs?wSDL</code></p> <p>where:</p> <p><code>webservice</code></p> <p>Is the name of the Web service you created using Application Explorer.</p> <p>To find this value, navigate to the Integration Business Services tab, expand the <i>Services</i> node, select the service you created, and click the <i>Service Description</i> link on the right. The WSDL URL appears in the Address field of the window that opens.</p> <p>You can also open the WSDL file in a third party XML editor (for example, XMLSPY) and view the SOAP request settings to find this value.</p>
<code>soapaction</code>	<p>The method that will be called by the SOAP disposition.</p> <p>This value is found in the WSDL file.</p>
<code>method</code>	<p>The Web service method you are using. This value is found in the WSDL file.</p>
<code>namespace</code>	<p>The XML namespace you are using. This value is found in the WSDL file.</p>
<code>responseTo</code>	<p>The location to which responses are posted. A predefined port name or another full URL. Optional. The URL must be complete, including the protocol.</p>
<code>errorTo</code>	<p>The location to which error logs are sent. Optional.</p> <p>A predefined port name or another disposition URL. The URL must be complete, including the protocol.</p>

5. Click OK.

In the left pane, the event port appears under the ports node. In the right pane, summary information associated with the event port you created appears. An example of a port listing and summary information is shown in the following image.



Use the scroll bar at the bottom of the Operations pane to see the entire Disposition data.

You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-21.

Procedure: How to Create an Event Port for HTTP

To create an event port for an HTTP disposition using Application Explorer:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node and the *Oracle Applications* node.
3. Select the *ports* node.

The following image shows the ports node selected on the left and the Operations menu open on the right



4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane, shown in the following image, opens on the right with four fields to identify the new port, a Help button, and two action buttons.

Create New Port

Choose parameters of the port that you wish to create.

Port Name:

Description:

Disposition Protocol:

Disposition:

- a. In the Port Name field, type a name for the event port.
- b. In the Description field, type a brief description of the port.
- c. From the Disposition Protocol drop-down list, select *HTTP*.
- d. In the Disposition field, enter an HTTP destination.

When pointing Application Explorer to an **ibSE** deployment, specify the destination file using the following format:

`ihttp://url[;responseTo=respDest]`

When pointing Application Explorer to a **JCA** deployment, specify the destination file using the following format:

`http://host:port/uri`

The following table lists and describes the disposition parameters for HTTP.

Parameter	Description
<code>url</code>	URL target for the post operation.

Parameter	Description
<i>respDest</i>	Location where responses are posted. A predefined port name or another full URL. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.
<i>host</i>	Name of the host on which the Web server resides.
<i>port</i>	Port number on which the Web server is listening.
<i>uri</i>	Universal resource identifier that completes the url specification.

5. Click **OK**.

In the left pane, the event port appears under the ports node. In the right pane, summary information associated with the event port you created appears. An example of a port listing and summary information is shown in the following image.



You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-21.

Procedure: How to Create an Event Port for MQSeries

To create an event port for an MQSeries queue using Application Explorer:

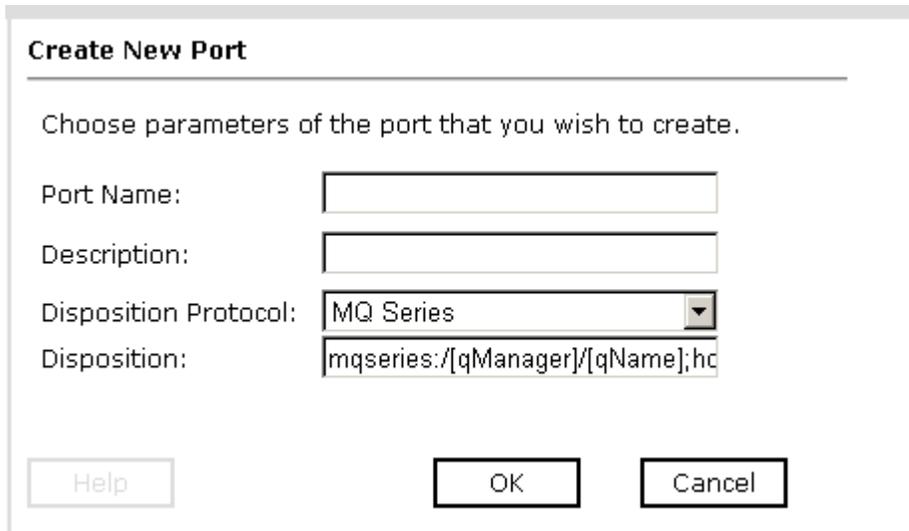
1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node and the *Oracle Applications* node.
3. Select the *ports* node.

The following image shows the ports node selected on the left and the Operations menu open on the right



4. In the right pane, move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane, shown in the following image, opens on the right providing four fields to identify the new port.



- a. In the Port Name field, type a name for the event port.
- b. In the Description field, type a brief description of the port.
- c. In the Disposition Protocol drop-down list, select *MQSeries*.
- d. In the Disposition field, enter an MQSeries destination.

When pointing Application Explorer to an **ibSE** deployment, specify the destination file using the following format:

```
mqseries: /qManager/qName;host=hostName;port=portNum  
;channel=channel[;errorTo=errorDest]
```

When pointing Application Explorer to a **JCA** deployment, use the following format:

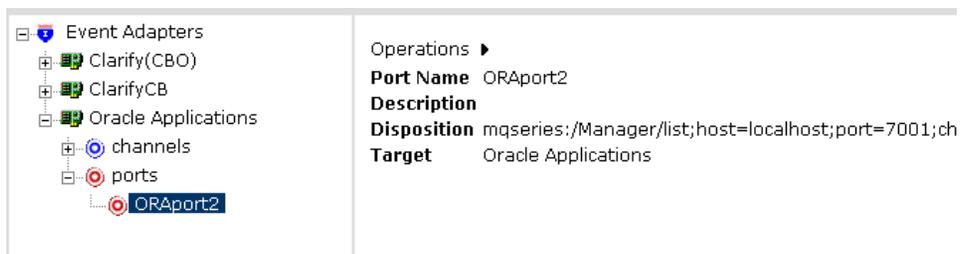
```
mq:qmanager@respqueue;host=;port=;channel=
```

The following table lists and describes the disposition parameters for MQSeries.

Parameter	Description
<i>qManager</i>	Name of the queue manager to which the server must connect.
<i>qName or respqueue</i>	Name of the queue where messages are placed.
<i>hostName</i>	Name of the host on which MQSeries resides (MQ client only).
<i>portNum</i>	The port number for connecting to an MQ Server queue manager (MQ client only).
<i>channel</i>	Case-sensitive name of the channel that connects with the remote MQ Server queue manager (MQ client only). The default MQ Series channel name is SYSTEM.DEF.SVRCONN.
<i>errorDest</i>	Location where error documents are sent. This can be a predefined port name or another full URL. Optional.

5. Click **OK**.

In the left pane, the event port appears under the ports node. In the right pane, summary information associated with the event port you created appears. An example of a port listing and summary information is shown in the following image.



Use the scroll bar at the bottom of the Operations pane to see the entire Disposition data.

You are now ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 3-21.

Editing or Deleting an Event Port

The following procedures describe how to edit and delete an event port using Application Explorer.

Procedure: How to Edit an Event Port

To edit an existing event port using Application Explorer:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and the *ports* node.
3. Select the event port you want to edit.
4. In the right pane, move the pointer over *Operations* and select *Edit*.

The Edit Port pane, shown in the following image, opens on the right with fields to modify the name, description, disposition protocol, and disposition of the port.

Edit Port

Choose parameters of the port that you wish to edit.

Port Name:

Description:

Disposition Protocol:

Disposition:

5. Make the required changes and click *OK*.

Procedure: How to Delete an Event Port

To delete an existing event port using Application Explorer:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and the *ports* node.

3. Select the event port you want to delete.
4. In the right pane, move the pointer over *Operations* and select *Delete*.
A confirmation dialog box opens.
5. To delete the event port, click *OK*.
The event port disappears from the list in the left pane.

Creating a Channel

The following procedure describes how to create a channel for an Oracle E-Business Suite event. All defined event ports must be associated with a channel.

Procedure: How to Create a Channel

To create a channel using Application Explorer:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node and the *Oracle Applications* node.
3. Select the *channels* node.
4. In the right pane, move the pointer over *Operations* and select *Add a new channel*.

The Add a new ORACLE APPLICATIONS channel pane, shown in the following image, opens on the right. This pane includes fields to enter a name and description and a drop-down list from which to select a channel type.

Add a new ORACLE APPLICATIONS channel

Choose a name and description for the new channel that you wish to create.

Channel Name:

Description:

Channel Type:

Help < Back Next > Cancel

- a. In the Channel Name field, type a name for the channel.

- b. In the Description field, type a brief description of the channel.
 - c. From the Channel Type drop-down list, accept the default channel type, *TABLE Listener*.
5. Click *Next*.

The Edit channels pane, shown in the following image, opens on the right. The Edit channels pane provides three tabs for Thin Driver, OCI Driver, and Run Time Properties. Each tab displays parameter fields related to the specific driver and properties.

The image shows a dialog box titled "Edit channels". It has three tabs: "Thin Driver", "OCI Driver" (which is selected and highlighted in green), and "Run Time Properties". Below the tabs are five input fields labeled "Host:", "Port:", "SID:", "User:", and "Password:". At the bottom of the dialog are four buttons: "Help", "< Back", "Next >", and "Cancel".

6. Enter the parameter values for either the Thin Driver, the OCI Driver, or Run Time Properties. The tables found in this section define the parameters of these drivers.

If your Oracle system uses the Oracle JDBC Thin Driver, select the *Thin Driver* tab and specify the information from the following table. The table lists and describes the fields for the JDBC Thin Driver.

Field	Description
Host	Name of the server on which the Oracle E-Business Suite database instance resides.
Port	Port number on which the database is listening.

Field	Description
SID	Unique name of the database service, chosen by the database administrator or the person who installed Oracle E-Business Suite.
User	Oracle database user ID to access the Oracle database underlying the Oracle E-Business Suite system. The user ID must have database access to the interface tables being accessed.
Password	Password associated with the specified user ID.

If your Oracle system uses the Oracle JDBC OCI Driver, select the *OCI Driver* tab and specify the information from the following table that lists and describes the fields for the JDBC OCI Driver.

Field	Description
Connection TNS Name	Logical name of the Oracle E-Business Suite database instance. (You can obtain this from the database administrator.) The TNS name that is found in the tnsnames.ora file that contains an entry for that database and that <i>resides on the server running the adapter</i> . If a TNS name does not exist, you must create it. For information about creating a TNS name, see your Oracle Administrator.
Release	Release number of the Oracle E-Business Suite instance. Select it from the list of releases supported by the adapter.
User	Oracle database user ID to access the Oracle database underlying the Oracle E-Business Suite system. The user ID must have database access to the interface tables being accessed.
Password	Password associated with the specified user ID.

If you select the Run Time Properties tab, edit the information in the Run Time Properties tab using the field descriptions in the following table.

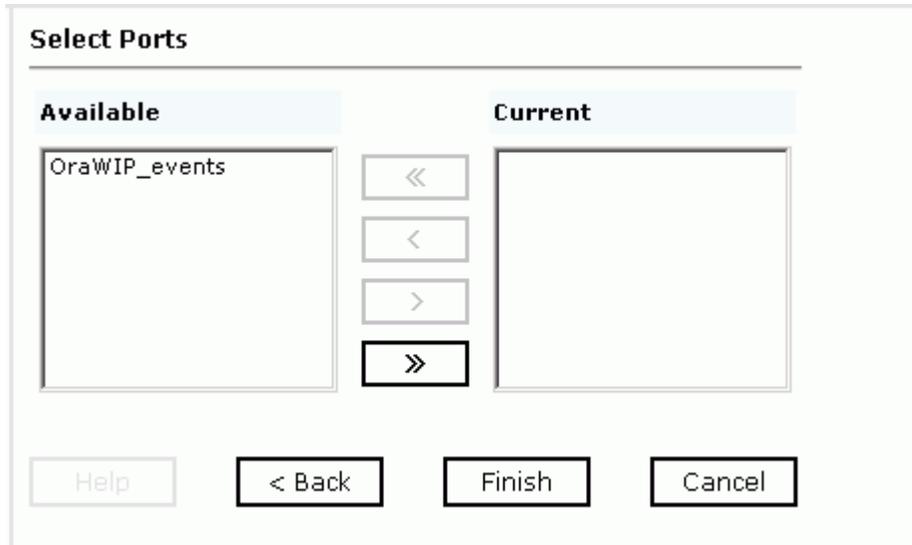
Field	Description
Polling Interval (ms)	Interval, in milliseconds, at which to check for new input.

Field	Description
SQL Query	<p>The SQL SELECT statement that the listener issues to poll the table.</p> <p>If the statement is omitted, it defaults to</p> <pre>SELECT * FROM table</pre> <p>where:</p> <pre>table</pre> <p>Is the value of the Node Name property.</p> <p>If the SQL statement includes a date column or long text column, you must provide a value for the Post Query property. The value you provide must not contain a date column or a long text column. This applies whether you provide an SQL statement here or rely upon the default.</p> <p>For example, the following SELECT statement retrieves all unprocessed records from the DISCRETE_JOBS table:</p> <pre>SELECT * FROM WIP_DISCRETE_JOBS DJ WHERE DJ.WIP_ENTITY_ID > (SELECT WIP_ENTITY_ID FROM WIP.TEMP_NEW_WORK_ORDER_ENTITY_ID)</pre>

Field	Description
Post Query*	<p>One or more SQL statements that are executed after each new record is read from the table. Separate multiple statements with a semicolon (;).</p> <p>Case sensitive. The case used to specify the column names must match the case used in the SELECT statement that polled the table. If the SQL Query property was omitted so that a default SELECT statement polled the table, the case used to specify the column names must match the case used to define the columns in the database's schema.</p> <p>*This property and Issue Post Query Delete are mutually exclusive: you must select one or the other, but not both.</p> <p>When you specify a value for this parameter, the table data is retained after it is read.</p> <p>If you do not specify a value for Post Query, you must select the Issue Post Query Delete property, which specifies that the adapter deletes each record after it is read. (If you want to retain the table's data after it is read, specify a value for Post Query, and do not select Issue Post Query Delete.)</p> <p>There are two field operators, ? and ^, that you can use in a post-query SQL statement. For more information, see <i>The Post Query Parameter Operators</i> on page 3-28.</p>
Node Name	Name of the database table to query with the default SELECT statement if the SQL Query property is omitted. This is ignored if SQL Query is specified.
Issue Post Query Delete*	<p>If you select this option, the adapter deletes each record after it is read.</p> <p>*This property and Post Query are mutually exclusive: you must select one or the other, but not both.</p>

7. Click *Next*.

The Select Ports pane, shown in the following image, opens on the right. This pane provides lists of available and current ports and buttons to enable you to move ports from one list to the other.



8. Associate one or more ports with this channel.
 - To associate one port, select a port from the list of current ports and click the single right arrow button to transfer the port to the list of available ports. You can repeat this to associate additional ports.
 - To associate all ports, click the double right arrow button.
9. Click *Finish*.

The summary window opens.

A summary provides the channel description, channel status, and available ports. All the information is associated with the channel you created.

The channel appears under the channels node in the left pane.

An X over the icon indicates that the channel is currently disconnected. You must start the channel to activate your event configuration.

Procedure: How to Start a Channel

To start a channel:

1. Click the *Event Adapters* tab.

2. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and the *channels* node.
3. Select the channel you want to start.
4. In the right pane, move the pointer over *Operations* and select *Start the channel*.
The channel becomes active.
In the left pane, the X that was over the icon disappears.

To stop the channel at any time, move the pointer over *Operations* and select *Stop the channel*.

Procedure: How to Edit a Channel

To edit an existing channel:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and the *channels* node.
3. Select the channel you want to edit.
4. In the right pane, move the pointer over *Operations* and select *Edit*.
The Edit ORACLE APPLICATIONS channel pane opens.
5. Edit the Description field, if required, and click *Next*.
Another Edit channel pane opens.
6. Edit the Thin Drive, OCI Driver, or Run Time Properties, as required, and click *Next*.
The Selected Ports pane opens.
7. When you complete your changes, click *Finish*.

Procedure: How to Delete a Channel

To delete an existing channel:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and the *channels* node.
3. Select the channel you want to delete.
4. In the right pane, move the pointer over *Operations* and select *Delete*.
A confirmation dialog box opens.
5. To delete the channel you selected, click *OK*.

The channel disappears from the list in the left pane.

The Post Query Parameter Operators

You can use two special field operators, ? and ^, with the Post Query parameter. Both of these operators dynamically substitute database values in the SQL post-query statement at run time.

- *?fieldname* is evaluated at run time as *field = value*

The ? operator is useful in UPDATE statements:

```
UPDATE table WHERE ?field
```

For example, the following statement

```
UPDATE Stock_Prices_Temp WHERE ?RIC
```

might be evaluated at run time as:

```
UPDATE Stock_Prices_Temp WHERE RIC = 'PG'
```

- *^fieldname* is evaluated at run time as *value*

The ^ operator is useful in INSERT statements:

```
INSERT INTO table VALUES (^field1, ^field2, ^field3, ...)
```

For example, the following statement

```
INSERT INTO Stock_Prices_Temp VALUES (^RIC, ^Price, ^Updated)
```

might be evaluated at run time as:

```
INSERT INTO Stock_Prices_Temp VALUES ('PG', 88.62, '2003-03-18  
16:24:00.0')
```

Choosing a Listening Technique

You can detect an Oracle E-Business Suite event by using an RDBMS Table Listener for Oracle E-Business Suite. The Table Listener polling technology enables you to specify SQL SELECT statements to execute periodically. After data is polled, it passes through the adapter for additional processing.

You can poll a relational or non-relational database directly and send the results to a file or JMS message queue. You use the following techniques to listen to an Oracle E-Business Suite event:

- **Standard event processing with row tracking.** The listener polls a table, sends each newly inserted row to a destination you specify (known as the disposition), and uses a control table to keep track of which row was most recently read. The control table prevents the most recently read row from being reread during the next listening cycle.

You can apply this flexible yet simple technique in most situations.

For more information, see *Standard Event Processing With Row Tracking* on page 3-30.

- **Standard event processing with row removal.** The listener polls a table, sends each newly inserted row to a destination you specify, and then deletes the new row from the table to prevent it from being reread during the next listening cycle.

Apply this technique when the source table is being used to pass data to the adapter, and the table's rows do not need to persist. Rows will be deleted as they are processed.

For more information, see *Standard Event Processing With Row Removal* on page 3-33.

- **Trigger-based event processing.** At design time you assign triggers to a joined group of tables. At run time the triggers write information about table changes to a common control table; the listener polls the control table and sends information about the table changes to a destination you specify. The listener deletes new rows from the control table to prevent them from being reread during the next listening cycle.

Apply this technique when listening to events in a group of large joined tables, or when you need to know if a row has been updated or deleted.

For more information, see *Trigger-based Event Processing* on page 3-35.

Standard Event Processing With Row Tracking

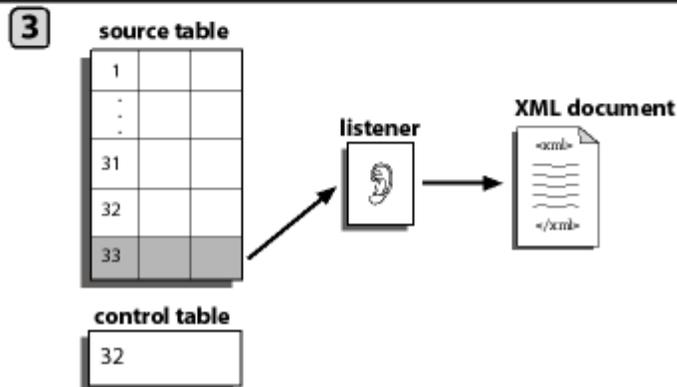
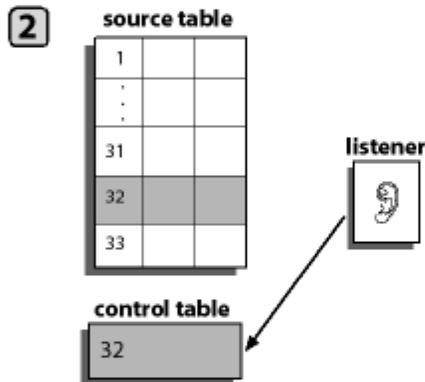
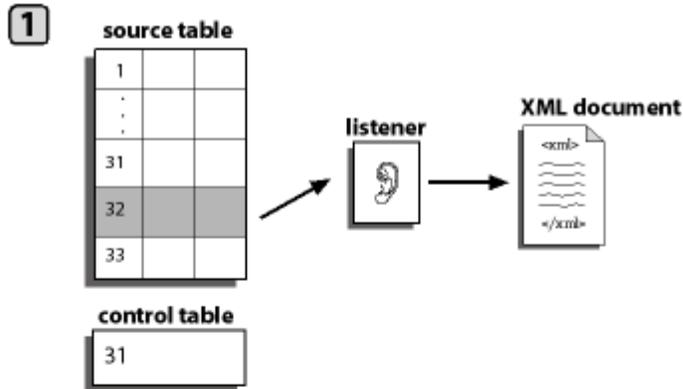
The standard event processing with row tracking technique enables you to listen to the source table without removing its rows. It requires you to create a single-cell control table that keeps track of the last new row the RDBMS Table Listener for Oracle E-Business Suite read from the source table.

The control table's one column corresponds to a column (or to a group of columns) in the source table that is unique, sortable, and indicates how recently the row was added to the source table relative to the other rows. For example, the first row added to the source table has the lowest value, and the last row added has the highest value. This value is the "event key."

When you create the control table, initialize it to the event key of the row most recently added to the source table. When you specify the listener's properties, configure the listener's Post Query property to automatically update the control table's event key.

Each time the listener queries the source table, it looks for rows added since the last query—that is, for rows whose event key is greater than the current value of the field in the control table. It reads each row of this type and returns it to the specified destination using an XML document. To ensure that the row is not read again the next time the listener queries the table, the listener updates the field in the control table to match the value of the row just read from the source table.

The following figure illustrates standard event processing with row tracking. It shows the source and control tables, the listener, and the XML document at various stages.



In the previous figure:

1. The Table Listener queries the source table and copies each source table row whose event key is greater than the control table's event key. The listener copies the row to an XML document and sends it to the destination defined in the port disposition.
2. The listener updates the event key in the control table to match the row it has most recently read.
3. The listener copies the next source table row to an XML document. The process repeats.

To implement this event processing technique, see *How to Implement Standard Event Processing With Row Tracking* on page 3-32.

Procedure: How to Implement Standard Event Processing With Row Tracking

To implement standard event processing with row tracking:

1. Create a control table. For an example, see *Creating the Control Table for an Oracle E-Business Suite Event* on page 3-32.
2. Configure an RDBMS Table Listener for Oracle E-Business Suite using Application Explorer.

In addition to the required listener properties, for standard event processing with row tracking you must also provide values for the following optional properties:

- **SQL Query**, the SQL SELECT statement that identifies the source table to which the adapter listens, and with which it queries the table.
- **Post Query**, the SQL statements that maintain the field in the control table.

For instructions for configuring a listener, see *Creating a Channel* on page 3-21.

Example: Creating the Control Table for an Oracle E-Business Suite Event

Follow the steps in this example to create a table named TEMP_NEW_YORK_ORDER_ENTITY that has a single field named WIP_ENTITY_ID. You specify this table when you configure the RDBMS Table Listener for Oracle E-Business Suite, as described in *Creating a Channel* on page 3-21.

When discrete jobs are created through the Oracle E-Business Suite graphical interface, an entry is created in the WIP.WIP_DISCRETE_JOBS table. For this example, you configure an event to detect new entries to this table. You will use the standard event processing with row tracking technique. (Oracle E-Business Suite processing cannot delete rows from the table.) To accomplish this, first create a simple table to keep track of the records processed.

1. From within Oracle SQL*PLUS, run the following SQL:

```
CREATE TABLE WIP.TEMP_NEW_WORK_ORDER_ENTITY_ID
(
  WIP_ENTITY_ID NUMBER
)
```

This creates a single table with a single field.

Note: Oracle SQL*Plus is part of the Oracle client software. If it is not installed, contact your Oracle Database Administrator.

You must be logged in under the APPS schema or a similar ID that has access rights to the Oracle E-Business Suite WIP schema.

2. Create a single record in this table and seed it with the highest WIP_ENTITY_ID ID from your system. You can obtain this from the WIP.WIP_DISCRETE_JOBS table.

This sets the value at which to start detecting events as records enter the WIP_DISCRETE_JOBS table.

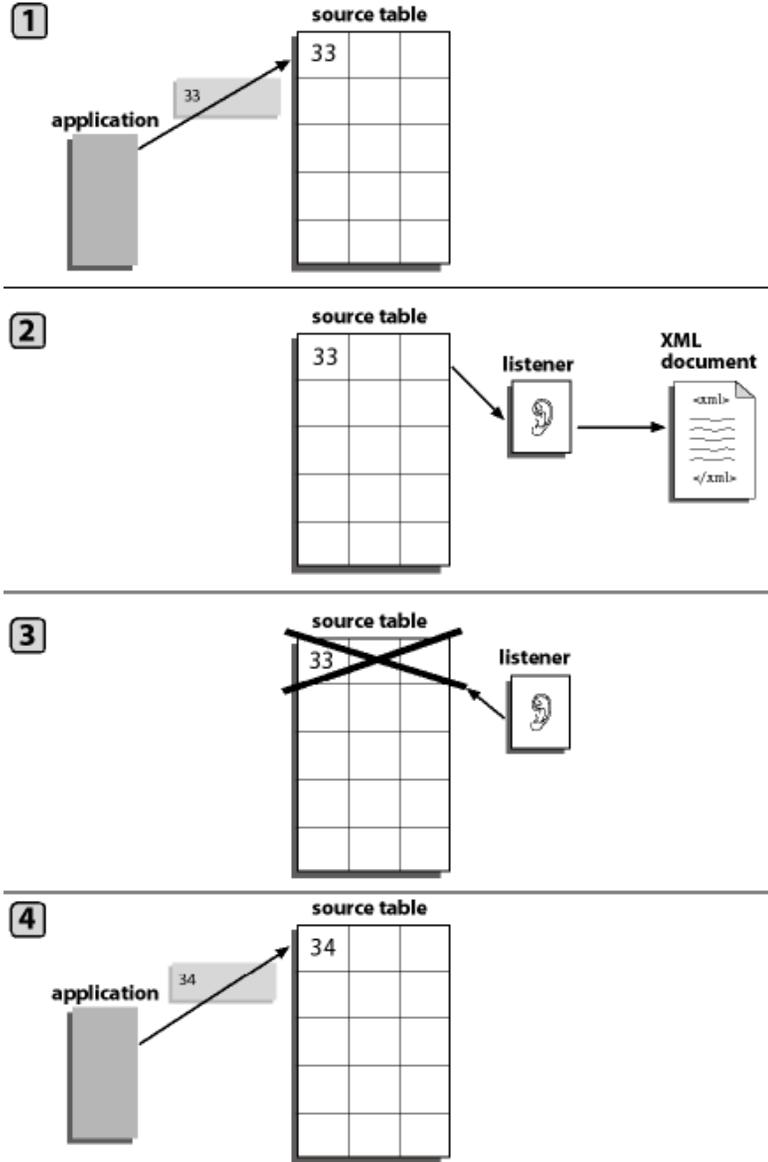
After you create a simple table in Oracle, you must configure the Table Listener, as described in *Creating a Channel* on page 3-21.

Standard Event Processing With Row Removal

The standard event processing with row removal technique assumes that the source table is being used as a conduit to pass the data to the adapter, and that the table's rows do not need to persist. The RDBMS Table Listener for Oracle E-Business Suite periodically queries the source table: when it finds a row, it reads it and returns it to the Reply_to destination via an XML document. To ensure that the row is not read again when the Table Listener next queries the table, the listener then deletes the row from the table.

Choosing a Listening Technique

The following figure illustrates standard event processing with row removal. It shows the application, source table, listener, and the XML document at various stages in the event processing.



In the previous figure:

1. Your application inserts a new row into the source table.
2. The listener queries the source table and copies the new row to an XML document and sends it to the destination defined in the port disposition using the File protocol.
3. The listener deletes the source table row to ensure that the row is not read again when the listener next queries the table.
4. The application inserts a new row into the source table.
5. The process repeats itself.

To implement this event processing technique, see *How to Implement Standard Event Processing With Row Removal* on page 3-35.

Procedure: How to Implement Standard Event Processing With Row Removal

To implement the standard event processing with row removal technique:

1. Configure an RDBMS Table Listener.
2. In addition to the required listener properties, provide values for the following optional properties:
 - **SQL Query**, the SQL SELECT statement that identifies the source table to which the adapter will listen, and with which it will query the table.
 - **Issue Post Query Delete**, which automatically deletes each record after it has been read.

For detailed instructions for configuring a listener, see *Creating a Channel* on page 3-21.

Trigger-based Event Processing

Trigger-based event processing is a technique for listening to multiple joined Oracle E-Business Suite tables. It is also helpful for detecting when a row has been deleted or updated.

The trigger-based technique provides the following benefits:

- **Improved performance when listening to events in a group of large joined tables.**

When processing joined tables, Oracle creates a Cartesian product working table. When the joined tables are large, the interim working table is *very* large. The standard technique of processing Oracle E-Business Suite events, in which the adapter periodically listens to the entire structure of joined tables, can consume a significant amount of computing resources.

The trigger-based technique avoids this overhead by requiring the RDBMS Table Listener for Oracle E-Business Suite to query a single small control table and by writing to the control table only when an event actually occurs.

- **Increased number of event types that the adapter recognizes.**

Using the trigger-based technique, you can tell when a row was updated, deleted, or inserted. Using the standard technique, you can tell only when a row was inserted.

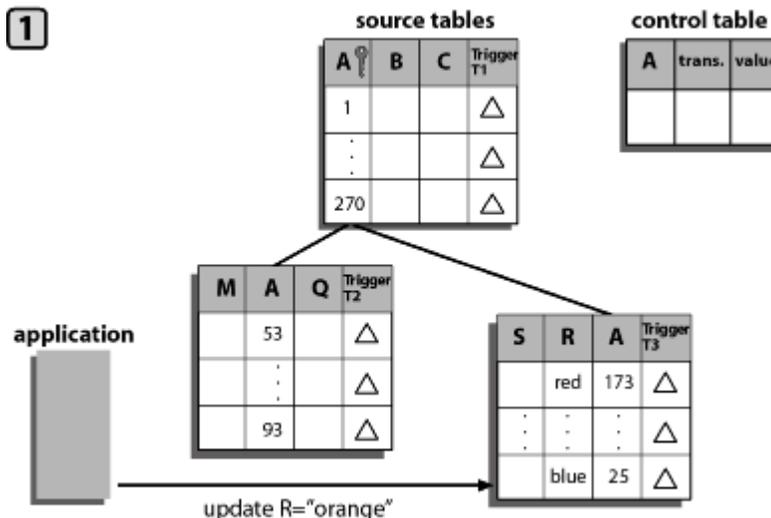
To use the trigger-based technique, you assign a trigger to each table that you want to monitor. When a value changes, it fires the corresponding trigger, which writes data to a control table. The BEA WebLogic Adapter for Oracle E-Business Suite listens to this control table by running a query against it. When it finds a row in the control table, it reads it and returns it to the port disposition created when the port is configured via an XML document. To ensure that the row is not read again when the listener next queries the table, the listener then deletes the row from the table.

The trigger-based technique enables you to recognize changes to an entity. For the purposes of this discussion, an entity is a real-world object that is represented in the database by a hierarchical set of tables.

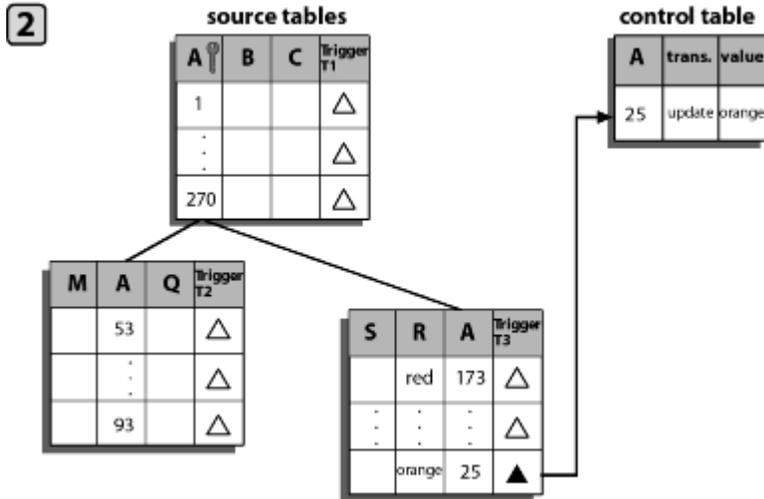
You manage the triggers using SQL*Plus or a similar tool, and configure the listener using Application Explorer.

The following series of five figures illustrates the steps involved in trigger-based event processing. It shows the source and control tables, the listener, the application, and the XML document at various stages in the event processing.

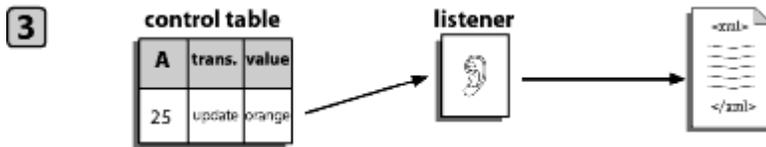
1. Your application updates a row in a group of related source tables



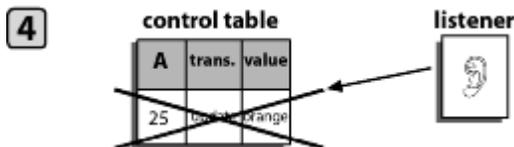
- The update causes a row trigger to fire in the changed table. The trigger inserts a row into the control table. The new control table row includes the key value (25), the type of transaction (update), and the new cell value (orange).



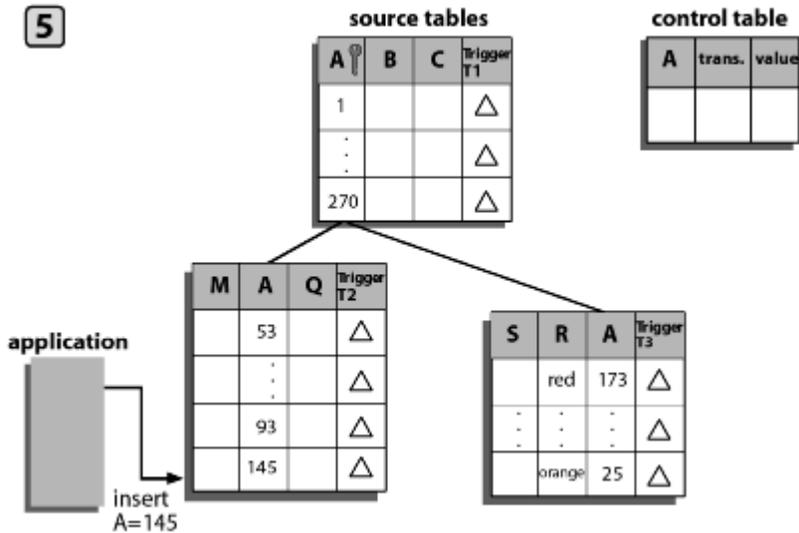
- The listener queries the control table and copies the new row to an XML document. It sends the document to the destination defined in the port disposition.



- The listener deletes the control table row to ensure that the row is not read again when the listener next queries the table.



- The application inserts a new row into one of the source tables. The process repeats itself.



For a summary of how to implement this technique, see *How to Implement Trigger-based Event Processing* on page 3-38.

Procedure: How to Implement Trigger-based Event Processing

To implement the trigger-based event processing technique:

- Create the control table.

The purpose of the control table is to capture the key of each entity that changed, regardless of which of the entity tables changed.

You can store a variety of information in the control table, including the key of the entity that was inserted, updated, or deleted, and the name of the table and field that was updated.

The design of the control table is a function of the business logic of your application. For example, you can choose between creating one control table for a group of joined source tables or one control table per source table. Among the issues to consider are the kinds of events to monitor (insertions, deletions, and/or updates), and whether you want to monitor only the highest-level table in a group of joined tables or all of the tables in the group.

2. Assign triggers to the source tables.

The triggers you assign, and to which tables you assign them, is determined by what kind of change you want to monitor. The triggers implement much of the event-processing logic. For a sample trigger, see *Trigger on WIP_ENTITY_NAME Column* on page 3-40.

For example, consider a bill of material scenario. (A bill of materials is a list of all the parts required to manufacture an item, the subparts required for the parts, and so on. The complete item/parts/subparts relationship can extend to several levels, creating a data structure like a tree with the finished item as the root.) In a bill of materials, where each level in the parts hierarchy is represented by a separate table, you might assign a trigger to only the highest-level table (the finished product), or you might assign triggers to all tables (the finished product and its parts and subparts).

As another example, if multiple changes are made to the same row during one listener cycle, you could configure the event adapter to record all the changes. If a row was inserted and then updated, both changes would be logged.

3. Configure the RDBMS Table Listener for Oracle E-Business Suite when creating a channel using Application Explorer.

In addition to the required listener properties, for trigger-based event processing you also must provide values for the following optional properties:

- **SQL Query**, the SQL SELECT statement that identifies the control table to which the adapter listens, and with which it queries the table to determine changes in the source tables.
- **Post Query**, to identify the rows that the adapter automatically deletes from the control table.

For detailed instructions for configuring a listener, see *Creating a Channel* on page 3-21.

Example: Trigger on WIP_ENTITY_NAME Column

The following trigger fires when a change is made to the WIP_ENTITY_NAME column of the WIP.WIP_ENTITIES table. When it fires, it writes the relevant values to the control table IWAY.IWAY_PO_CDC.

```
CREATE OR REPLACE TRIGGER IWAY.IWAY_PO_CDC_WE_TRG

AFTER INSERT OR DELETE OR UPDATE OF WIP_ENTITY_NAME
ON WIP.WIP_ENTITIES
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    INSERT INTO IWAY.IWAY_PO_CDC
      VALUES (
        :NEW.WIP_ENTITY_ID,
        :NEW.ORGANIZATION_ID,
        'UPDATE' );

  ELSE
    INSERT INTO IWAY.IWAY_PO_CDC
      VALUES (
        :OLD.WIP_ENTITY_ID,
        :OLD.ORGANIZATION_ID,
        'UPDATE' );

  END IF;

EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    NULL;          -- Record already exists

END;
```

CHAPTER 4

Using Web Services Policy-Based Security

Topics:

- iWay Business Services Policy-Based Security
- Configuring iWay Business Services Policy-Based Security

Servlet Application Explorer provides a security feature called iWay Business Services policy-based security. The following topics describe how this feature works and how to configure it.

Note: For the iWay 5.5 RG2 Release, it is recommended that policy-based security not be enabled.

iWay Business Services Policy-Based Security

iWay Business Services provide a layer of abstraction between the back-end business logic they invoke and the user or application running the business service. This enables easy application integration but raises the issue of controlling the use and execution of critical and sensitive business logic that is run as a business service.

Servlet Application Explorer controls the use of business services that use adapters with a feature called policy-based security. This feature enables an administrator to apply *policies* to iWay Business Services (iBS) to deny or permit their execution.

A *policy* is a set of privileges associated with the execution of a business service that can be applied to an existing or new iBS. When you assign specific rights or privileges inside a policy, you need not recreate privileges for every iBS that has security issues in common with other iWay Business Services. Instead, you can use one policy for many iWay Business Services.

The goal is to secure requests at both the transport and the SOAP request level that is transmitted on the wire. Some policies do not deal with security issues directly but affect the run-time behavior of the business services to which they are applied.

The iBSE administrator creates an instance of a policy type, names it, associates individual users and/or groups (a collection of users), and then applies the policy to one or more business services.

You can assign a policy to an iBS or to a method within an iBS. If a policy is applied only to a method, other methods in that iBS are not governed by it. However, if a policy is applied to the iBS, all methods are governed by it. At run time, the user ID and password that are sent to iBSE in the SOAP request message are checked against the list of users for all policies applied to the specific iBS. The Resource Execution policy type is supported and dictates who can or cannot execute the iBS.

When a policy is not applied, the default value for an iBS is to “grant all.” For example, anyone can execute the iBS until the Resource Execution policy is associated to the iBS. At that time, only users granted execution permission, or those who do not belong to a group that was denied execution permissions, have access to the iBS.

Configuring iWay Business Services Policy-Based Security

Before you create instances of policies, you must have a minimum of one user or one group to associate to an instance. You can create users and groups using Servlet Application Explorer. For more information, see *Create a User to Associate With a Policy* on page 4-3 or *Create a Group to Associate With a Policy* on page 4-5.

An execution policy governs who can execute the business service to which the policy is applied. For more information, see *Create an Execution Policy* on page 4-8.

You configure the IP and Domain Restriction policy type slightly differently from other policy types. The IP and Domain Restriction policy type controls connection access to iBSE and therefore, need not be applied to an individual business service. You need not create a policy, however, you must enable the Security Policy option in Servlet Application Explorer. For more information, see *Configure IP and Domain Restrictions* on page 4-12.

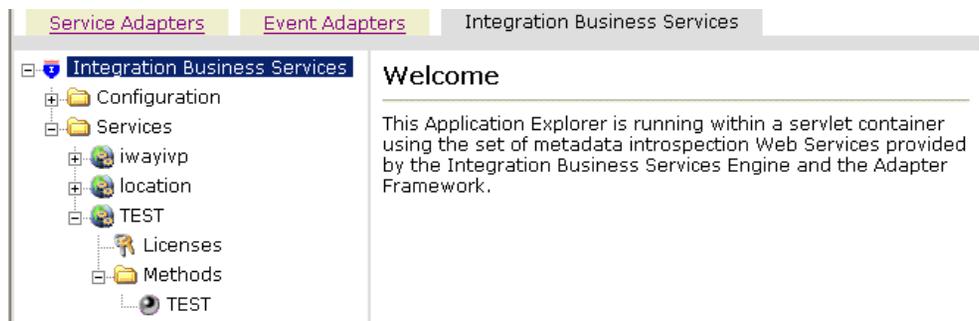
Note: For the iWay 5.5 RG2 Release, it is recommended that policy-based security not be enabled.

Procedure: How to Create a User to Associate With a Policy

To create a user to associate with a policy:

1. Open *Servlet Application Explorer*.

The following image shows the window that opens and includes three tabs corresponding to Service Adapters, Event Adapters, and iWay Business Services. The iWay Business Services tab is active and displays a Welcome screen on the right. The image shows the iWay Business Services node expanded in the left pane.



- a. Click the *iWay Business Services* tab.
- b. Expand the *Configuration* node.
- c. Expand the *Security* node.
- d. Expand the *Users and Groups* node.

- e. Select *Users*.
2. In the right pane, move the pointer over *Operations* and select *Add*.

The following image shows the Add a new user pane that opens and includes fields where you enter a user name, a password, and a description of the user. The pane includes a Help button, an OK button to instruct the system to accept inputs, and a Cancel button to escape from the pane.

Add a new user

Name:

Password:

Description:

- a. In the Name field, type a user ID.
 - b. In the Password field, type the password associated with the user ID.
 - c. In the Description field, type a description of the user (optional).
3. Click *OK*.

The following image opens and shows a new user added to the configuration. It includes a definition of a user and a user ID and description.

Operations ▶



Users

A user is an object that can be granted or denied permissions to run Integration Business Services. A user can belong to one or more groups. Policies that specify particular rights can be associated with user.

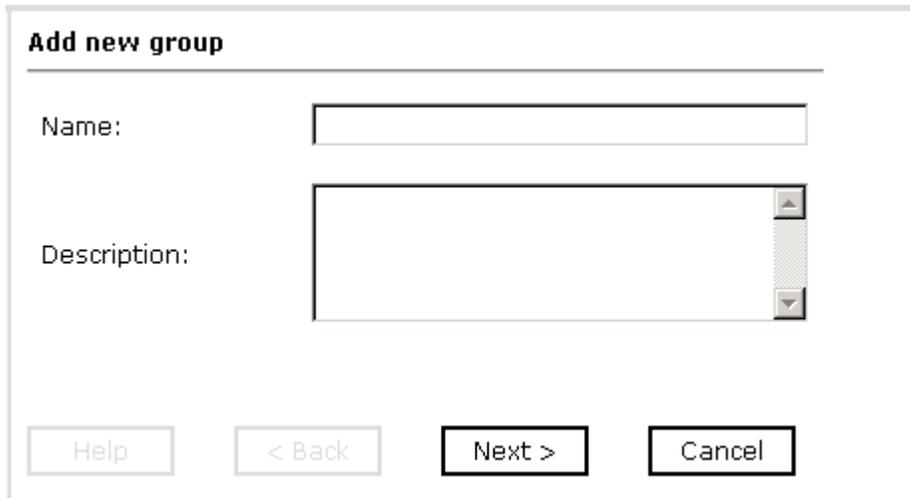
User Id	Description
<input type="checkbox"/> bse1	

Procedure: How to Create a Group to Associate With a Policy

To create a group to associate with a policy:

1. Open *Servlet Application Explorer*.
 - a. Click the *iWay Business Services* tab.
 - b. Expand the *Configuration* node.
 - c. Expand the *Security* node.
 - d. Expand the *Users and Groups* node.
 - e. Select *Groups*.
2. In the right pane, move the pointer over *Operations* and click *Add*.

The following image shows the Add new group pane that opens with fields where you enter a name and a description for the group. To continue after typing inputs, click the Next button. The pane also includes a Help button, a Back button to return to the previous screen, and a Cancel button to escape from the pane.



The image shows a dialog box titled "Add new group". It contains two input fields: "Name:" with a single-line text box and "Description:" with a multi-line text area. At the bottom, there are four buttons: "Help", "< Back", "Next >", and "Cancel". The "Next >" button is highlighted with a black border.

- a. In the Name field, type a name for the group.
 - b. In the Description field, type a description for the group (optional).
3. Click Next.

The following image shows the Modify Group Membership pane where you can move users to or from a group using the arrow keys to move them between the Current and Available lists and then clicking the Finish button. The pane includes a Help button, a Back button to return to the previous screen, and a Cancel button to escape from the pane.

Modify Group Membership

The screenshot shows a window titled "Modify Group Membership". It is divided into two main sections: "Current" and "Available". The "Current" section is an empty rectangular box. The "Available" section contains the text "bse1". Between these two sections is a vertical column of four arrow buttons: a double left arrow (←←), a single left arrow (←), a single right arrow (→), and a double right arrow (→→). Below the "Current" and "Available" sections are four buttons: "Help", "< Back", "Finish", and "Cancel".

You can either highlight a single user in the list of available users and add it to the current list by clicking the left arrow, or you can click the double left arrow to add all users in the list of available users to the group.

4. After you select a minimum of one user, click *Finish*.

The new group is added.

The following image shows a pane with a new group added to the configuration. It includes a definition of a group and the group name and description.

Operations ▶



Groups

A group is an object that can be granted or denied permissions to run Integration Business Services. A group is used as a container for one or more users. Policies that specify particular rights can be associated with a group.

Group name	Description
------------	-------------

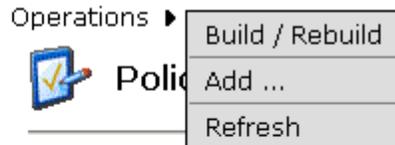
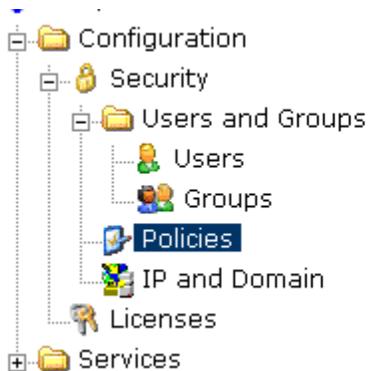
<input type="checkbox"/> newgroup	
-----------------------------------	--

Procedure: How to Create an Execution Policy

To create an execution policy:

1. Open *Servlet Application Explorer*.
 - a. Click the *iWay Business Services* tab.
 - b. Expand the *Configuration* node.
 - c. Select *Policies*.

The following image shows the Policies pane on the right where you apply a policy. The Operations menu becomes available with three options, Build/Rebuild, Add, and Refresh.



You can configure policies for the to manage resource execution, : and failover/recovery actions.

2. Move the pointer over *Operations* and click *Add*.

The following image shows the Add a new policy pane that opens with fields for entering the name, type, and description of the policy. To continue, click the Next button. The pane includes a Help button, a Back button to return to the previous screen, and a Cancel button to escape from the pane.

Add a new policy

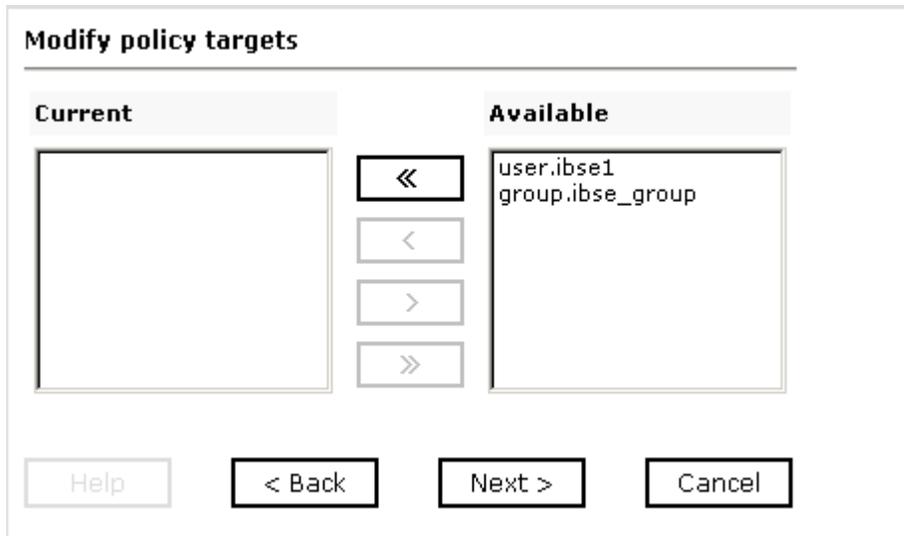
Name:

Type:

Description:

- a. In the Name field, type a name for the policy.
 - b. From the Type drop-down list, select *Execution*.
 - c. In the Description field, type a description for the policy (optional).
3. Click *Next*.

The following image shows the Modify policy targets pane that opens and includes a list of current and available targets and arrow buttons to move targets from one list to the other. The pane also includes a Help button, a Back button to return to the previous screen, a Next button to continue to the next screen, and a Cancel button to escape from the pane.



4. Select a minimum of one user or group from the Available pane.

Note: This user ID is checked against the value in the user ID element of the SOAP header sent to iBSE in a SOAP request.

5. Click *Next*.

The following image shows the Modify policy permissions pane that opens and includes drop-down lists where you can select to grant or deny permission to members and then click a button to finish. The pane also includes a Help button, a Back button to return to the previous screen, and a Cancel button to escape from the pane.

Member Id	Permission
user.ibse1	Deny
group.ibse_group	Deny

Buttons: Help, < Back, Finish, Cancel

6. To assign whether users or groups may execute the iBSE, select *Grant* to permit execution or *Deny* to restrict execution from a Permission drop-down list.
7. Click *Finish*.

The following image shows the pane that summarizes your configuration. It includes a definition of policies and the name, type, and description of the policies.

Operations ▶

Policies

You can configure policies for the Integration Business Services Engine to manage resource execution, service routing, data restrictions and failover/recovery actions.

Name	Type	Description
<input type="checkbox"/> ibse_policy	Execution	

Procedure: How to Configure IP and Domain Restrictions

To configure IP and domain restrictions:

1. Open *Servlet Application Explorer*.
 - a. Select the *iWay Business Services* tab.
 - b. Expand the *Configuration* node.
 - c. Expand the *Security* node.
 - d. Select *IP and Domain*.
2. In the right pane, move the pointer over *Operations* and click *Add*.

The following image shows the Add a new IP/Domain pane that opens where you enter information for the IP/Domain in four fields. You must select a type of restriction from a drop-down list before you can enter information in the IP(Mask)/Domain field. The pane also includes a Help button, an OK button to instruct the system to accept inputs, and a Cancel button to escape from the pane.

Add a new IP/Domain

IP(Mask)/Domain:

Type:

Access Control:

Description:

- a. From the Type drop-down list, select the type of restriction.
- b. In the IP(Mask)/Domain field, type the IP or domain name using the following guidelines.

If you select **Single (Computer)** from the **Type** drop-down list, you must provide the IP address for that computer. If you only know the DNS name for the computer, click *DNS Lookup* to obtain the IP Address based on the DNS name.

If you select **Group (of Computers)**, you must provide the IP address and subnet mask for the computer group.

If you select **Domain**, you must provide the domain name, for example, yahoo.com.

3. From the **Access Control** drop-down list, select *Grant* to permit access or *Deny* to restrict access for the IP addresses and domain names you are adding.
4. Click **OK**.

The following image shows the pane that opens and summarizes your configuration including the domain name, whether access is granted or denied, and a description (optional).

Operations ▶



IP and Domain

You can configure the Integration Business Services Engine to use policies that control access from a single IP address, a group of IP addresses, or all addresses within a particular domain.

IP(Mask) / Domain	Access	Description
<input type="checkbox"/> test	Deny	

CHAPTER 5

Management and Monitoring

Topics:

- Managing and Monitoring Services and Events Using iBSE
- Managing and Monitoring Services and Events Using the JCA Test Tool
- Setting Engine Log Levels
- Configuring Connection Pool Sizes
- Migrating Repositories
- Exporting or Importing Targets
- Retrieving or Updating Web Service Method Connection Information
- Starting or Stopping a Channel Programmatically

After you create services and events using Servlet Application Explorer, you can use managing and monitoring tools provided by the Integration Business Services Engine (iBSE) and the iWay Connector for JCA to measure the performance of your run-time environment. This section describes how to configure and use these features.

Managing and Monitoring Services and Events Using iBSE

Integration Business Services Engine (iBSE) provides a console to manage and monitor services and events currently in use and to display resource usage and invocation statistics. These indicators can help you adjust your environment for optimum efficiency.

The following monitoring levels are available for services:

- System
- Service
- Method

The following monitoring levels are available for events:

- System
- Channel
- Port

Procedure: How to Configure Monitoring Settings

To configure monitoring settings:

1. Ensure that your BEA WebLogic Server is started.
2. To access the monitoring console, enter the following URL in your Web browser:

`http://localhost:port/ibse/IBSEConfig`

where:

`localhost`

Is the machine where the application server is running.

`port`

Is the HTTP port for the application server.

The following image shows the iBSE Settings window that opens. It lists property names and includes fields where you can enter values for each property. To configure system settings, the System pane contains drop-down lists for selecting language, encoding, the debug level, and the number of asynchronous processors. It also contains a field where you can enter a path to the adapters lib directory.

To configure security settings, the Security pane contains fields for typing the Admin User name and the associated password and a check box for specifying policy.

To configure repository settings, the Repository pane, as shown in the following image, contains a drop-down list for selecting the repository type, fields to type information for the repository URL, driver, user, and password, and a check box where you can enable repository pooling. In the upper and lower right of the window is a Save button. In the lower left of the window is an option to click to access more configuration settings.

iBSE Settings:		Save
Property Name	Property Value	
System		
Language	English ▾	
Adapter Lib Directory	C:\Program Files\WWay55\lib	
Encoding	UTF-8 ▾	
Debug Level	NONE ▾	
Number of Async. Processors	0 ▾	
Security		
Admin User	jway	
Admin Password	****	
Policy	<input type="checkbox"/>	
Repository		
Repository Type	File System ▾	
Repository Url	file://C:\Program Files\WWay55\bea\ibs	
Repository Driver		
Repository User		
Repository Password		
Repository Pooling	<input type="checkbox"/>	
More configuration...		
		Save

3. Click *More configuration*.

Tip: To access the monitoring console directly, enter the following URL in your Web browser:

<http://localhost:port/ibse/IBSEStatus>

where:

[localhost](#)

Is the machine where the application server is running.

[port](#)

Is the HTTP port for the application server.

The following image shows the iBSE Monitoring Settings window that opens. It lists property names and includes a corresponding field where you can enter values for each property. The Monitoring pane contains a drop-down list for selecting the repository type, fields to type information for the repository URL, driver, user, and password, and a check box where you can enable repository pooling. The Auditing pane contains an option button to click to specify whether to store a message and a drop-down list where you can select the maximum messages to store. At the bottom of the window is a row of buttons that you can click to save your configuration, view events, or view services. The Save History button is inactive. After you enter properties and choose whether to save or view, you can click the Start Monitoring button.

Property Name	Property Value
Monitoring	
Repository Type	File System
Repository Url	file://C:\Program Files\W\ay55\bea
Repository Driver	
Repository User	
Repository Password	
Repository Pooling	<input type="checkbox"/>
Auditing	
Store Message	<input type="radio"/> yes <input checked="" type="radio"/> no
Max Message Stored	10,000
Save Configuration Save History View Events View Services	
Start Monitoring	

- a. In the Monitoring pane, from the Repository Type drop-down list, select the type of repository you are using.
- b. To connect to the database in the Repository Url field, type a JDBC URL.
- c. To connect to the database in the Repository Driver field, type a JDBC Class.
- d. To access the monitoring repository database, type a user ID and password.
- e. To enable pooling, click the *Repository Pooling* check box.
- f. In the Auditing pane, select *yes* if you want to store messages.

This option is disabled by default.

Note: You must start and then, stop monitoring to enable this option.

- g. Select the maximum number of messages you want to store.

By default, 10,000 is selected.

Note: Depending on your environment and the number of messages that are exchanged, storing a large number of messages may affect system performance. If you need more information about your system resources, consult your system administrator.

- h. Click *Save Configuration*.
4. Click *Start Monitoring*.
iBSE begins to monitor all services and events currently in use. If you selected the option to store messages, iBSE stores messages.
5. To stop monitoring, click *Stop Monitoring*.

Procedure: How to Monitor Services

To monitor services:

1. Ensure that your BEA WebLogic Server is started.
2. From the iBSE Monitoring Settings window, click *Start Monitoring*.
3. Click *View Services*.

The following image shows the System Level Summary (Service Statistics) window that opens. The Web Service Methods pane contains a drop-down list where you select a service. On the right, space is reserved for a drop-down list of methods that will appear. The Statistics pane contains a table with a summary of service statistics and two drop-down lists where you can select a successful or failed invocation to view more information about that service. At the bottom of the window is a home button to click to return to the iBSE Monitoring Settings window.

The screenshot shows a window titled "Service Statistics". It is divided into two main sections: "Web Service Methods" and "Statistics".

Web Service Methods

Service: Method:

Statistics

Total Time	55 min
Total Request Count	1
Total Success Count	1
Total Error Count	0
Average Request Size	409.0 bytes
Average Response Size	665.0 bytes
Average Execution Time	656 ms
Last Execution Time	828 ms
Average Back End Time	530 ms
Last Back End Time	765 ms
Successful Invocations	<input type="text" value="select a correlation id"/>
Failed Invocations	<input type="text" value="select a correlation id"/>

The system level summary provides services statistics at a system level.

The following table consists of two columns, one that lists the name of each statistic and the other that describes the corresponding service statistic.

Statistic	Description
Total Time	Total amount of time iBSE monitors services. The time starts after you click Start Monitoring in the iBSE Monitoring Settings window.
Total Request Count	Total number of services requests that were made during the monitoring session.
Total Success Count	Total number of successful service executions.
Total Error Count	Total number of errors that were encountered.
Average Request Size	Average size of an available service request.
Average Response Size	Average size of an available service response size.
Average Execution Time	Average execution time for a service.
Last Execution Time	Last execution time for a service.
Average Back End Time	Average back end time for a service.
Last Back End Time	Last back end time for a service.
Successful Invocations	A list of successful services arranged by correlation ID. To retrieve more information for a service, you can select the service from the drop-down list.
Failed Invocations	A list of failed services arranged by correlation ID. To retrieve more information for a service, you can select the service from the drop-down list.

4. Select a service from the drop-down list.

The following image shows the System Level Summary (Service Statistics) window that opens. The Web Service Methods pane contains a drop-down list on the left where you select a service and a drop-down list on the right where you select a service method. The Statistics pane contains a table with a summary of service statistics and two drop-down lists. To view more information about that service, you can select it from the Successful Invocations or Failed Invocations drop-down list. To suspend or resume a service, you can click a button in the lower right. To return to the iBSE Monitoring Settings window, you click the home button (also located in the lower right).

Service Statistics

Web Service Methods

Service	Method
<input type="text" value="B0100033"/>	<input type="text" value="all methods"/>

Statistics

Total Time	1 hrs
Total Request Count	1
Total Success Count	1
Total Error Count	0
Average Request Size	409.0 bytes
Average Response Size	665.0 bytes
Average Execution Time	656 ms
Last Execution Time	656 ms
Average Back End Time	530 ms
Last Back End Time	530 ms
Successful Invocations	<input type="text" value="select a correlation id"/>
Failed Invocations	<input type="text" value="select a correlation id"/>

- a. To stop a service at any time, click *Suspend Service*.
- b. To restart the service, click *Resume Service*.
5. Select a method for the service from the Method drop-down list.

The following image shows the Method Level Summary (Service Statistics) window that opens. The Web Service Methods pane contains a drop-down list on the left where you select a service and a drop-down list on the right where you select a service method. The Statistics pane contains a table with a summary of service statistics and two drop-down lists. To view more information about that service, you can select it from the Successful Invocations or Failed Invocations drop-down list. To suspend or resume a service, you can click a button in the lower right. To return to the iBSE Monitoring Settings window, you click the home button (also located in the lower right).

Service Statistics

Web Service Methods

Service	Method
<input type="text" value="B0100033"/>	<input type="text" value="GetEffectiveAddress"/>

Statistics

Total Time	1 hrs
Total Request Count	1
Total Success Count	1
Total Error Count	0
Average Request Size	409.0 bytes
Average Response Size	665.0 bytes
Average Execution Time	656 ms
Last Execution Time	656 ms
Average Back End Time	530 ms
Last Back End Time	530 ms
Successful Invocations	<input type="text" value="select a correlation id"/>
Failed Invocations	<input type="text" value="select a correlation id"/>

6. For additional information about a successful service and its method, select a service based on its correlation ID from the Successful Invocation drop-down list.

The following image shows the Invocation Level Statistics window that opens. The Message Information pane contains a table of information about the message. The Client Information pane contains a table of information about the client. The Detail pane contains a table that shows the size of the request and response messages, with options to click to view the respective XML documents. In the lower right of the window is a home button to click to return to the iBSE Monitoring Settings window.

The screenshot displays the 'Invocation Statistics' window. It is divided into three main sections: 'Message Information', 'Client Information', and 'Detail'. Each section contains a table of data. In the 'Detail' section, the 'Request Message' and 'Response Message' entries are highlighted in blue, indicating they are clickable links. A '< home' button is located in the bottom right corner of the window.

Message Information	
Received	2004-09-14 12:04:16.312
Sent to adapter	2004-09-14 12:04:16.406
Received from adapter	2004-09-14 12:04:16.936
Responded	2004-09-14 12:04:16.968
Status	SUCCESS

Client Information	
Client IP	127.0.0.1
Client Host Name	127.0.0.1
User Name	

Detail	
Message	Size
Request Message	409 bytes
Response Message	665 bytes

< home

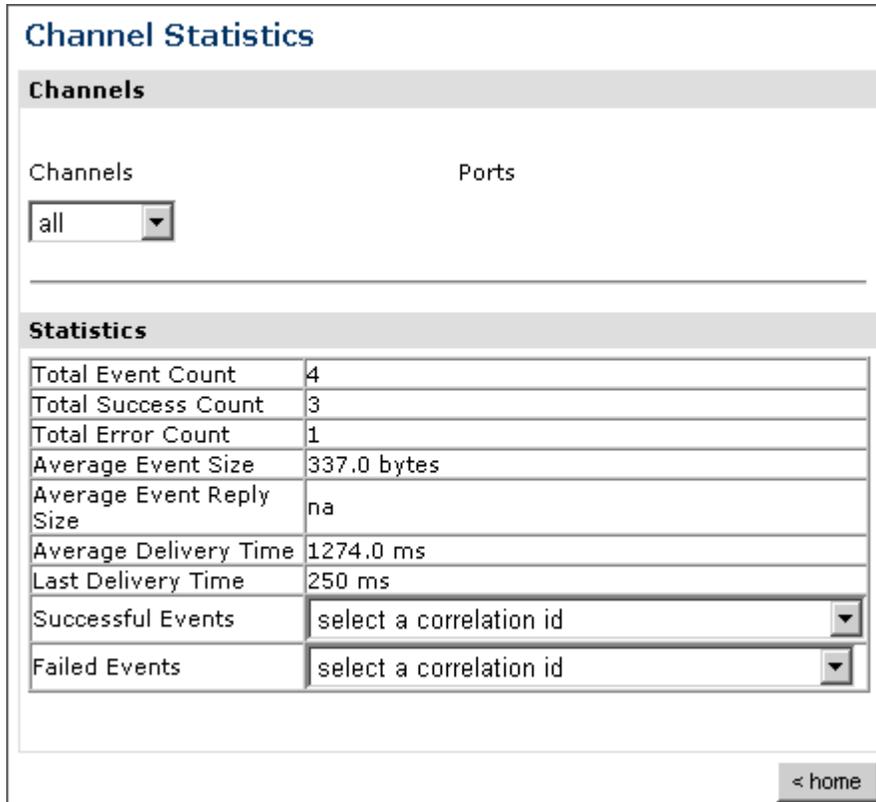
7. To view the XML request document in your Web browser, click *Request Message*.
You can also view the XML response document for the service.
8. To return to the iBSE Monitoring Settings window, click *home*.

Procedure: How to Monitor Events

To monitor events:

1. Ensure that your BEA WebLogic Server is started.
2. In the iBSE Monitoring Settings window, click *Start Monitoring*.
3. Click *View Events*.

The following image shows the System Level Summary (Channel Statistics) window that opens. The Channels pane contains a drop-down list on the left where you select a channel. On the right, space is reserved for a drop-down list of ports that will appear. The Statistics pane contains a table with a summary of event statistics and two drop-down lists where you can select a successful or failed event to view more information about that event. In the lower right of the window is a home button to click to return to the iBSE Monitoring Settings window.



The system level summary provides event statistics at a system level.

The following table consists of two columns, one that lists the name of each statistic and the other that describes the corresponding event statistic.

Statistic	Description
Total Event Count	Total number of events.
Total Success Count	Total number of successful event executions.

Statistic	Description
Total Error Count	Total number of errors that were encountered.
Average Event Size	Average size of an available event request.
Average Event Reply Size	Average size of an available event response.
Average Delivery Time	Average delivery time for an event.
Last Delivery Time	Last delivery time for an event.
Successful Events	List of successful events arranged by correlation ID. To retrieve more information for an event, select the event from the drop-down list.
Failed Events	List of failed events arranged by correlation ID. To retrieve more information for an event, select the event from the drop-down list.

4. Select a channel from the drop-down list.

The following image shows the Channel Level Event Summary (Channel Statistics) window that opens. The Channels pane contains a drop-down list on the left where you select a channel and a drop-down list on the right where you select a port. The Statistics pane contains a table with a summary of event statistics and two drop-down lists where you can select a successful or failed event to view more information about that event. In the lower right of the window is a button to click to suspend or resume a channel and a home button to click to return to the iBSE Monitoring Settings window.

Channel Statistics

Channels

Channels

Ports

Statistics

Total Event Count	3
Total Success Count	2
Total Error Count	1
Average Event Size	401.0 bytes
Average Event Reply Size	na
Average Delivery Time	1542.0 ms
Last Delivery Time	250 ms
Successful Events	<input type="text" value="select a correlation id"/>
Failed Events	<input type="text" value="select a correlation id"/>

Suspend Channel

Start Channel

- a. To stop a channel at any time, click *Suspend Channel*.
- b. To start the channel, click *Start Channel*.
5. From the Ports drop-down list, select a port for the channel.

The following image shows the Port Level Event Summary (Channel Statistics) window that opens. The Channels pane contains a drop-down list on the left where you select a channel and a drop-down list on the right where you select a port. The Statistics pane contains a table with a summary of event statistics and two drop-down lists where you can select a successful or failed event to view more information about that event. In the lower right of the window is a button to click to suspend or resume a channel and a home button to click to return to the iBSE Monitoring Settings window.

Channel Statistics

Channels

Channels: TestChan ▾ Ports: TestPort ▾

Statistics

Total Event Count	2
Total Success Count	2
Total Error Count	0
Average Event Size	446.0 bytes
Average Event Reply Size	na
Average Delivery Time	2189.0 ms
Last Delivery Time	na
Successful Events	select a correlation id ▾
Failed Events	select a correlation id ▾

Suspend Channel Start Channel

< home

6. For more information about a successful event and its port, select an event based on its correlation ID from the Successful Events drop-down list.

The following image shows the Event Level Statistics (Message Statistics) window that opens. The Message Information pane contains a table of information pertaining to the event message. The Messages pane contains a table that shows the size of the event and reply messages, with an option to view an XML document of the event message. In the lower right of the window is a home button to click to return to the iBSE Monitoring Settings window.

The screenshot displays the 'Message Statistics' window. It features two main sections: 'Message Information' and 'Messages'. The 'Message Information' section contains a table with three rows: 'Received At' (2004-09-14 12:18:20.842), 'Disposed At' (● TestPort), and 'Delivered At' (2004-09-14 12:18:23.562). The 'Messages' section contains a table with three rows: 'Detail' (size), 'Event Message' (446 bytes), and 'Reply Message' (na). A '< home' button is located in the bottom right corner of the window.

Message Information	
Received At	2004-09-14 12:18:20.842
Disposed At	● TestPort
Delivered At	2004-09-14 12:18:23.562

Messages	
Detail	size
Event Message	446 bytes
Reply Message	na

< home

- a. To view the XML event document in your Web browser, click *Event Message*.
- b. To return to the iBSE Monitoring Settings window, click *home*.

Managing and Monitoring Services and Events Using the JCA Test Tool

The JCA Test Tool, which is also known as the JCA Installation Verification Program (IVP), provides a console to manage and monitor services and events currently in use and to display resource usage and invocation statistics. These indicators can help you adjust your environment for optimum efficiency.

Procedure: How to Manage and Monitor Services Using the JCA Test Tool

To manage and monitor services using the JCA Test Tool:

1. Open a Web browser to:

<http://localhost:port/iwjcaivp>

where:

[localhost](#)

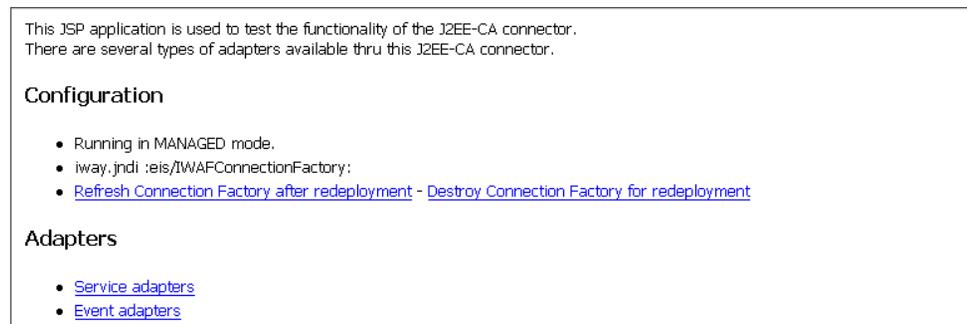
Is the name of the machine where your application server is running.

[port](#)

Is the port for the domain you are using. The port for the default domain is 7001.,for example:

<http://localhost:7001/iwjcaivp>

The following image shows the JCA Test Tool page that opens. The page contains a description of the function of the tool and configuration information, including options to change your connection settings. It also provides options for viewing service or event adapters.



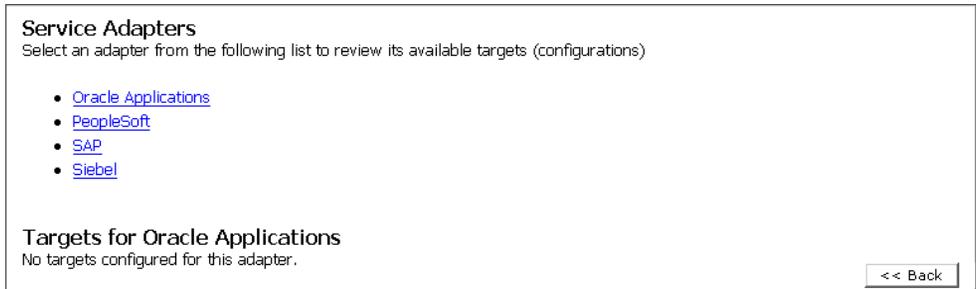
The JCA Test Tool runs in managed mode by default.

2. Perform the following steps to monitor the latest service adapter configuration.

Note: You must perform these steps for every new adapter target that is created using a JCA implementation of Application Explorer. In addition, you also must perform these steps for every new JCA configuration that is created using Application Explorer.

- a. Click *Destroy Connection Factory for redeployment*.
 - b. Redeploy the JCA connector module using the BEA WebLogic Server console.
 - c. In the JCA Test Tool, click *Refresh Connection Factory after redeployment*.
3. Click *Service adapters*.

The following image shows the Service Adapters page that opens. The page provides a live list of available service adapters and a list of targets configured for a specific adapter. In the lower right is a Back button to click to return to the previous page.



4. Select a service adapter to monitor.

The following image shows the page that opens. The left side provides a live list of available service adapters and a list of any targets configured for a specific adapter. The upper right side shows statistics for a selected target. The middle right has a User field and a Password field. The lower right contains a box where you type or paste an input document. Below the input box is a Send button to click to send a request for a test service and a Reset button to click to reset the fields. In the lower right is a Back button to click to return to the previous page.

The screenshot displays the JCA Test Tool interface. On the left, under "Service Adapters", there is a list of adapters: Oracle Applications, PeopleSoft, SAP, and Siebel. Below this is "Targets for Siebel" with a single target: TestService. On the right, "Statistics for Siebel target TestService" shows: TotalRequestCount: 0, TotalSuccessCount: 0, TotalErrorCount: 0, AverageExecutionTime: 0 msec, and LastExecutionTime: 0 msec. Below the statistics is the "Request for Siebel target TestService" section, which includes a note: "Enter the data for this interaction. The configured user/password will be used if the User name is not provided." This section contains three input fields: "User:", "Password:", and "Input Doc:". The "Input Doc:" field is a large text area. At the bottom of the form are three buttons: "Send", "Reset", and "<< Back".

- a. Click the desired target for your service adapter.
 - b. In the Request area, enter a user name and password.
 - c. In the Input Doc area, enter a request document that was created from the request schema for your service.
5. Click *Send*.

The following image shows the updated statistics that appear for your service if the request is successful. The statistics include the total number of requests, successes, and errors and the average and last execution time in milliseconds.

TotalRequestCount	: 0
TotalSuccessCount	: 0
TotalErrorCount	: 0
AverageExcecutionTime	: 0 msec.
LastExcecutionTime	: 0 msec.

Procedure: How to Manage and Monitor Events Using the JCA Test Tool

To manage and monitor events using the JCA Test Tool:

1. Open a Web browser to:

<http://localhost:port/iwjcaivp>

where:

[localhost](#)

Is the name of the machine where your application server is running.

[port](#)

Is the port for the domain you are using. The port for the default domain is 7001, for example:

<http://localhost:7001/iwjcaivp>

The following image shows the JCA Test Tool page that opens. The page contains a description of the function of the tool and configuration information, including options to change your connection settings. It also provides options for viewing service or event adapters.

This JSP application is used to test the functionality of the J2EE-CA connector. There are several types of adapters available thru this J2EE-CA connector.

Configuration

- Running in MANAGED mode.
- `iwjca.jndi :eis/IWAFConnectionFactory:`
- [Refresh Connection Factory after redeployment](#) - [Destroy Connection Factory for redeployment](#)

Adapters

- [Service adapters](#)
- [Event adapters](#)

The JCA Test Tool runs in managed mode by default.

2. Perform the following steps to monitor the latest event adapter configuration.

Note: You must perform these steps for every new adapter target that is created using a JCA implementation of Application Explorer. In addition, you must also perform these steps for every new JCA configuration that is created using Application Explorer.

- a. Click *Destroy Connection Factory for redeployment*.
 - b. Redeploy the JCA connector module using the BEA WebLogic Server console.
 - c. In the JCA Test Tool, click *Refresh Connection Factory after redeployment*.
3. Click *Event adapters*.

The Event Adapters page opens.

4. Select the event adapter to monitor.
5. Click the desired channel for your event adapter.
6. Click *start*.

The following image shows the updated statistics for your channel and the port. The statistics include the total number of requests, successes, and errors and the average and last execution time in milliseconds. There are options to click in the upper right of the page to start or refresh the channel.

Current channel Statistics	
Commands: start refresh	
Active: false	
TotalRequestCount	: 0
TotalSuccessCount	: 0
TotalErrorCount	: 0
AverageExcecutionTime	: 0 msec.
LastExcecutionTime	: 0 msec.
Statistics for port 'fileIN'	
TotalRequestCount	: 0
TotalSuccessCount	: 0
TotalErrorCount	: 0
AverageExcecutionTime	: 0 msec.
LastExcecutionTime	: 0 msec.

Setting Engine Log Levels

The following section describes how to set engine log levels for Servlet iBSE and JCA. For more information, see the *iWay Installation and Configuration for BEA WebLogic* documentation.

Procedure: How to Enable Tracing for Servlet iBSE

To enable tracing for Servlet iBSE:

1. Open the Servlet iBSE configuration page at:

`http://localhost:port/ibse/IBSEConfig`

where:

`localhost`

Is the name of the machine where your application server is running.

`port`

Is the port for the domain you are using. The port for the default domain is 7001, for example:

`http://localhost:7001/ibse/IBSEConfig`

2. In the System pane, from the Debug drop-down list, select the level of tracing.
3. Click *Save*.

The default location for the trace information on Windows is:

`C:\Program Files\bea\ibse\ibselogs`

Procedure: How to Enable Tracing for JCA

To enable tracing for JCA:

1. Open the extracted ra.xml file in a text editor.
2. Locate and change the following setting:

LogLevel. This setting can be set to DEBUG, INFO, or ERROR.

```
<context-param>
<config-property>
  <config-property-name>LogLevel</config-property-name>
  <config-property-type>java.lang.String</config-property-type>
  <config-property-value></config-property-value>
</config-property>
```

For example:

```
<config-property-value>DEBUG</config-property-value>
```

A directory in the configuration directory contains the logs.

- a. Review the logs generated by your application server.
 - b. Leave the remainder of the previous file unchanged.
3. Save the file and exit the editor.
 4. Redeploy the connector.

Configuring Connection Pool Sizes

The following topic describes how to configure connection pool sizes for the JCA connector.

Note: This section refers to connection pooling that is specific to the BEA WebLogic Adapter for SAP and the JCA Resource Connector within BEA WebLogic. The information is not related to SAP connection pooling.

Procedure: How to Configure Connection Pool Sizes

To configure connection pool sizes:

1. Open the extracted ra.xml file in a text editor.
2. Locate and change the following setting:

pool-params. The JCA Resource Connector has an initial capacity value of 0 by default and cannot be changed. The maximum capacity value is 10 by default and can be changed to a higher value.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE weblogic-connection-factory-dd (View Source for full
doctype...)>
- <weblogic-connection-factory-dd>
  <connection-factory-name>IWAFJCA</connection-factory-name>
  <jndi-name>eis/IWAFConnectionFactory</jndi-name>
  - <pool-params>
    <initial-capacity>0</initial-capacity>
    <max-capacity>10</max-capacity>
    <capacity-increment>1</capacity-increment>
    <shrinking-enabled>>false</shrinking-enabled>
    <shrink-period-minutes>200</shrink-period-minutes>
  </pool-params>
  <security-principal-map />
</weblogic-connection-factory-dd>
```

3. Save the file and exit the editor.
4. Redeploy the connector.

Migrating Repositories

During design time, a repository is used to store metadata created when using Application Explorer to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. For more information on configuring repositories, see the *iWay Installation and Configuration for BEA WebLogic* documentation.

The information in the repository also is referenced at run time. For management purposes, you can migrate iBSE and JCA repositories to new destinations without affecting your existing configuration. For example, you may want to migrate a repository from a development environment to a production environment. The BEA WebLogic Server must be restarted to detect new repository changes.

File Repositories

If you want to migrate a File repository to another destination, copy the `ibserepo.xml` file from the following path:

```
drive:\Program Files\iWay55\bea\ibse\ibserepo.xml
```

where:

```
drive
```

Is the location of your iWay 5.5 installation.

You can place the `ibserepo.xml` file in a new location that is a root directory of the iBSE Web application, for example:

```
drive:\ProductionConfig\bea\ibse\ibserepo.xml
```

iBSE Repositories

The following topic describes how to migrate an iBSE repository that is configured for Oracle. You can follow the same procedure if you want to migrate an iBSE repository that is configured for Microsoft SQL Server 2000, Sybase, or DB2. However, when you are configuring a new environment, you must execute the script that creates the repository tables for your database. In addition, verify that all required files and drivers for your database are in the class path. For more information on configuring repositories, see the *iWay Installation and Configuration for BEA WebLogic* documentation.

Note: The following procedure allows you to migrate only Web services. If migrating event handling information is one of your requirements, you must migrate at the database level. For more information, see *Migrating Event Handling Configurations* on page 5-28.

Procedure: How to Migrate an iBSE Repository Configured for Oracle

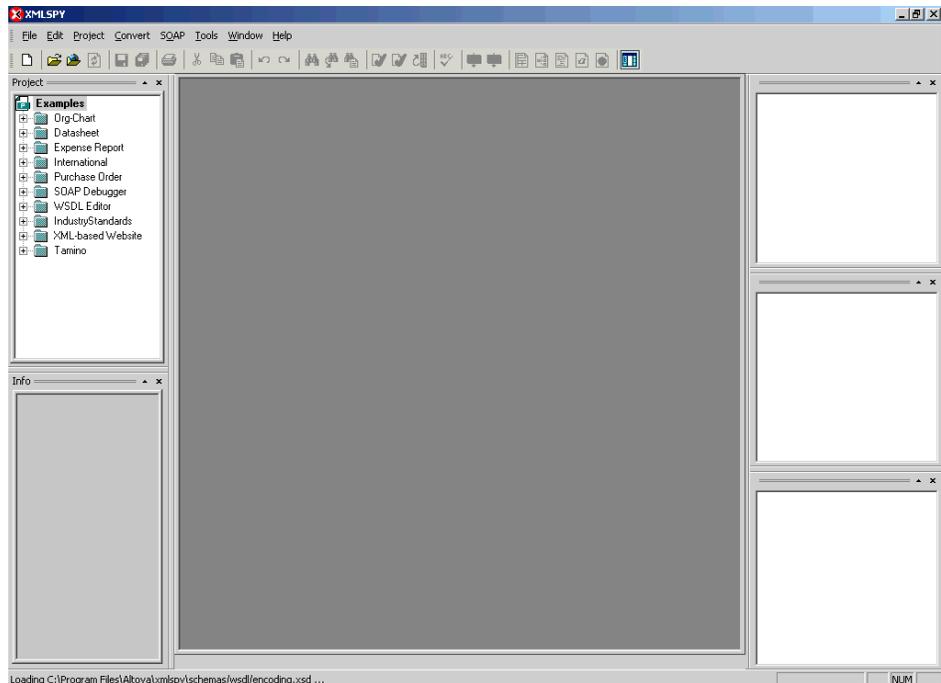
To migrate an iBSE repository that is configured for Oracle:

1. Copy the iBSE configuration service URL, for example:

<http://localhost:7777/ibse/IBSEServlet/admin/iwconfig.ibs?wsdl>

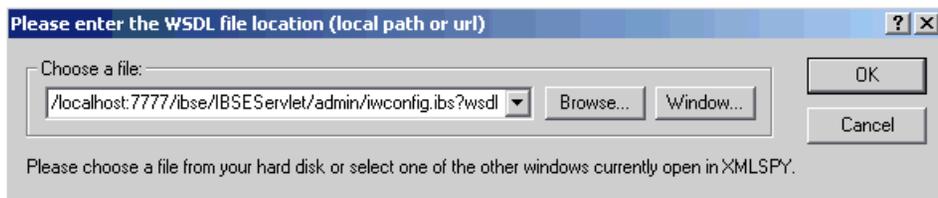
2. Open a third party XML editor, for example, XMLSPY.

The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.



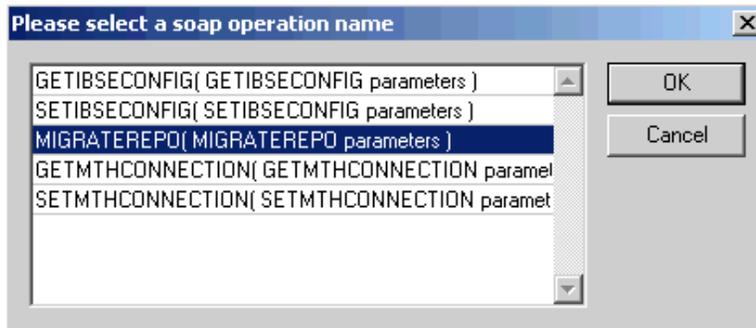
3. From the SOAP menu, select *Create new SOAP request*.

The following image shows the WSDL file location dialog box that opens, where you enter a local path or URL. The dialog includes Browse, Window, OK, and Cancel buttons.



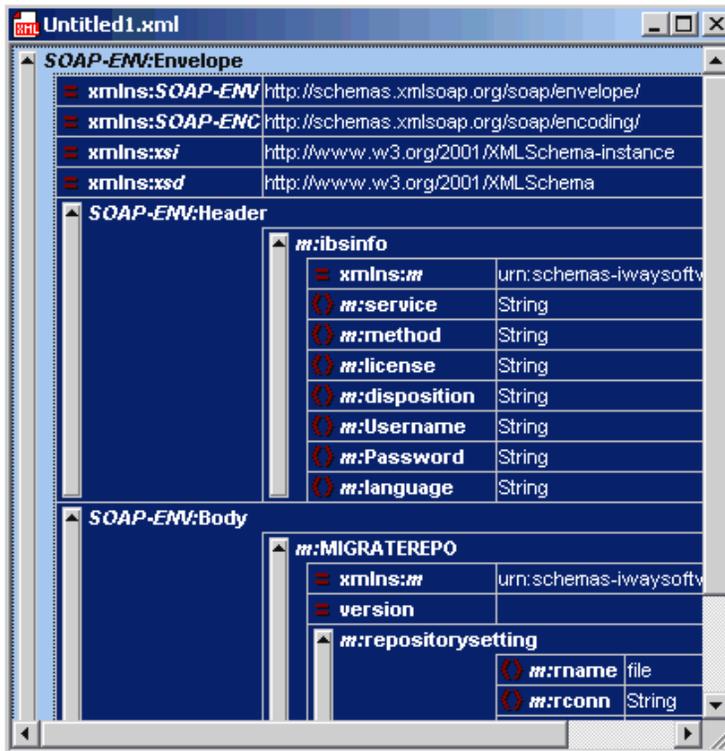
4. In the Choose a file field, paste the iBSE configuration service URL.
5. Click *OK*.

The following image shows the soap operation name dialog box that opens with a list of available control methods. You can select from the list and click *OK* or to escape from the dialog box, you can click *Cancel*.



6. Select the *MIGRATEREPO(MIGRATEREPO parameters)* control method and click *OK*.

The following image shows a portion of the window that opens with the structure of the SOAP envelope. It includes information about location and schemas.



7. Locate the *Text view* icon in the tool bar.

In the following image, the pointer points to the *Text view* icon.



8. To display the structure of the SOAP envelope as text, click the *Text view* icon.

The <SOAP-ENV:Header> tag is not required and can be deleted from the SOAP envelope.

9. Locate the following section:

```
<m:MIGRATEREPO
xmlns:m="urn:schemas-iwaysoftware-com:jul2003:ibse:config" version="">
<m:repositorysetting>
<m:rname>oracle</m:rname>
<m:rconn>String</m:rconn>
<m:rdriver>String</m:rdriver>
<m:ruser>String</m:ruser>
<m:rpwd>String</m:rpwd>
</m:repositorysetting>
<m:servicename>String</m:servicename>
</m:MIGRATEREPO>
```

- a. For the `<m:rconn>` tag, replace the String placeholder with the repository URL where you want to migrate your existing iBSE repository.

For example, the Oracle repository URL has the following format:

```
jdbc:oracle:thin:@[host]:[port]:[sid]
```

- b. For the `<m:rdriver>` tag, replace the String placeholder with the location of your Oracle driver.

Note: This is an optional tag. If you do not specify a value, the default Oracle JDBC driver is used.

- c. For the `<m:ruser>` tag, replace the String placeholder with a valid user name to access the Oracle repository.
- d. For the `<m:rpwd>` tag, replace the String placeholder with a valid password to access the Oracle repository.

10. Perform one of the following migration options.

If you want to migrate a **single** Web service from the current iBSE repository, enter the Web service name in the `<m:servicename>` tag, for example:

```
<m:servicename>Service1</m:servicename>
```

If you want to migrate **multiple** Web services from the current iBSE repository, duplicate the `<m:servicename>` tag for each Web service, for example:

```
<m:servicename>Service1</m:servicename>
<m:servicename>Service2</m:servicename>
```

If you want to migrate **all** Web services from the current iBSE repository, remove the `<m:servicename>` tag.

11. From the SOAP menu, select *Send request to server*.

Your iBSE repository and the Web services you specified migrate to the new Oracle repository URL that you specified.

JCA Repositories

The following procedure describes how to migrate a JCA repository. For more information on configuring JCA repositories, see the *iWay Installation and Configuration for BEA WebLogic* documentation.

Procedure: How to Migrate a JCA Repository

To migrate a JCA repository:

1. Navigate to the location of your JCA configuration directory where the repository schemas and other information is stored, for example:

```
C:\Program Files\iway55\config\base
```

2. Locate and copy the *repository.xml* file.
3. Place this file in a new JCA configuration directory to migrate the existing repository.

Your JCA repository migrates to the new JCA configuration directory.

Migrating Event Handling Configurations

This topic describes how to migrate your iBSE repositories at a database level for Microsoft SQL Server 2000, Oracle, Sybase, or DB2. You can use this information to migrate event handling information, for example, port or channel configurations.

Procedure How to Migrate a Microsoft SQL Server 2000 Repository

To migrate a Microsoft SQL Server 2000 repository:

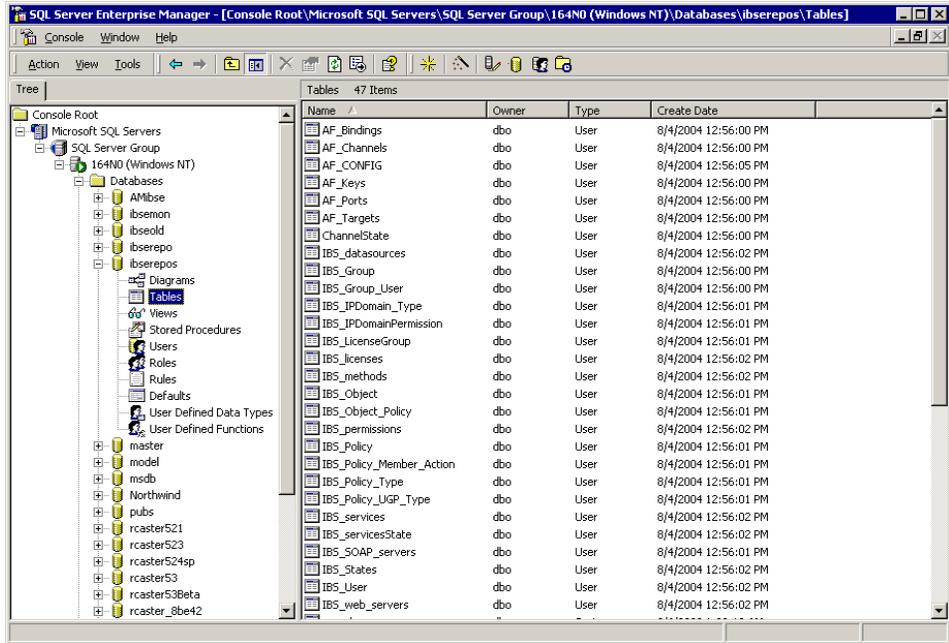
1. Open a command prompt and navigate to the iWay setup directory. The default location on Windows is:

```
C:\Program Files\iWay55\etc\setup
```

This directory contains SQL to create the repository tables in the following file:

```
iwse.sql
```

You can use `iwse.sql` to create the database tables that are used by iBSE. For example, the following image shows the tree in the left pane and tables in the right pane. The tables are listed by name in one column with corresponding columns for information about owner, type, and the date the table was created.



For more information on configuring the Microsoft SQL Server 2000 repository, see the *iWay Installation and Configuration for BEA WebLogic* documentation.

2. To migrate the tables that were created by the `iwse.sql` script for iBSE, use your Microsoft SQL Server 2000 database tool set. For more information, consult your database administrator.

Procedure How to Migrate an Oracle Repository

To migrate an Oracle repository:

1. Open a command prompt and navigate to the iWay setup directory. The default location on Windows is:

```
C:\Program Files\iWay55\etc\setup
```

This directory contains SQL to create the repository tables in the following files:

For Oracle 8:

```
iwse.ora
```

For Oracle 9:

[iwse.ora9](#)

2. To create the Oracle database tables that are used by iBSE, use the SQL script as shown in the example in the following image that shows a list of tables.



A screenshot of a list of Oracle database tables. Each table name is preceded by a small icon representing a table. The list includes:

- AF_Bindings
- AF_Channels
- AF_CONFIG
- AF_Keys
- AF_Ports
- AF_Targets
- ChannelState
- IBS_datasources
- IBS_Group
- IBS_Group_User
- IBS_IPDomain_Type
- IBS_IPDomainPermission
- IBS_LicenseGroup
- IBS_licenses
- IBS_methods
- IBS_Object
- IBS_Object_Policy
- IBS_permissions
- IBS_Policy
- IBS_Policy_Member_Action
- IBS_Policy_Type
- IBS_Policy_UGP_Type
- IBS_services
- IBS_servicesState
- IBS_SOAP_servers
- IBS_States
- IBS_User
- IBS_web_servers

For more information on configuring the Oracle repository, see the *iWay Installation and Configuration for BEA WebLogic* documentation.

3. To migrate the tables that were created by the SQL script for iBSE, use your Oracle database tool set. For more information, consult your database administrator.

Procedure How to Migrate a Sybase Repository

To migrate a Sybase repository:

1. Open a command prompt and navigate to the iWay setup directory. The default location on Windows is:

`C:\Program Files\iWay55\etc\setup`

This directory contains SQL to create the repository tables in the following file:

`sybase-iwse.sql`

2. To create the Sybase database tables that are used by iBSE, use the SQL script as shown in the example in the following image that shows a list of tables.



AF_Bindings
 AF_Channels
 AF_CONFIG
 AF_Keys
 AF_Ports
 AF_Targets
 ChannelState
 IBS_datasources
 IBS_Group
 IBS_Group_User
 IBS_IPDomain_Type
 IBS_IPDomainPermission
 IBS_LicenseGroup
 IBS_licenses
 IBS_methods
 IBS_Object
 IBS_Object_Policy
 IBS_permissions
 IBS_Policy
 IBS_Policy_Member_Action
 IBS_Policy_Type
 IBS_Policy_UGP_Type
 IBS_services
 IBS_servicesState
 IBS_SOAP_servers
 IBS_States
 IBS_User
 IBS_web_servers

For more information on configuring the Sybase repository, see the *iWay Installation and Configuration for BEA WebLogic* documentation.

3. To migrate the tables that were created by the SQL script for iBSE, use your Sybase database tool set. For more information, consult your database administrator.

Procedure How to Migrate a DB2 Repository

To migrate a DB2 repository:

1. Open a command prompt and navigate to the iWay setup directory. The default location on Windows is:

```
C:\Program Files\iWay55\etc\setup
```

This directory contains SQL to create the repository tables in the following file:

```
db2-iwse.sql
```

Exporting or Importing Targets

2. To create the DB2 database tables that are used by iBSE, use the SQL script as shown in the example in the following image that shows a list of tables.



- AF_Bindings
- AF_Channels
- AF_CONFIG
- AF_Keys
- AF_Ports
- AF_Targets
- ChannelState
- IBS_datasources
- IBS_Group
- IBS_Group_User
- IBS_IPDomain_Type
- IBS_IPDomainPermission
- IBS_LicenseGroup
- IBS_licenses
- IBS_methods
- IBS_Object
- IBS_Object_Policy
- IBS_permissions
- IBS_Policy
- IBS_Policy_Member_Action
- IBS_Policy_Type
- IBS_Policy_UGP_Type
- IBS_services
- IBS_servicesState
- IBS_SOAP_servers
- IBS_States
- IBS_User
- IBS_web_servers

For more information on configuring the DB2 repository, see the *iWay Installation and Configuration for BEA WebLogic* documentation.

You can migrate the tables that were created by the SQL script for iBSE using your DB2 database toolset. For more information, consult your database administrator.

Exporting or Importing Targets

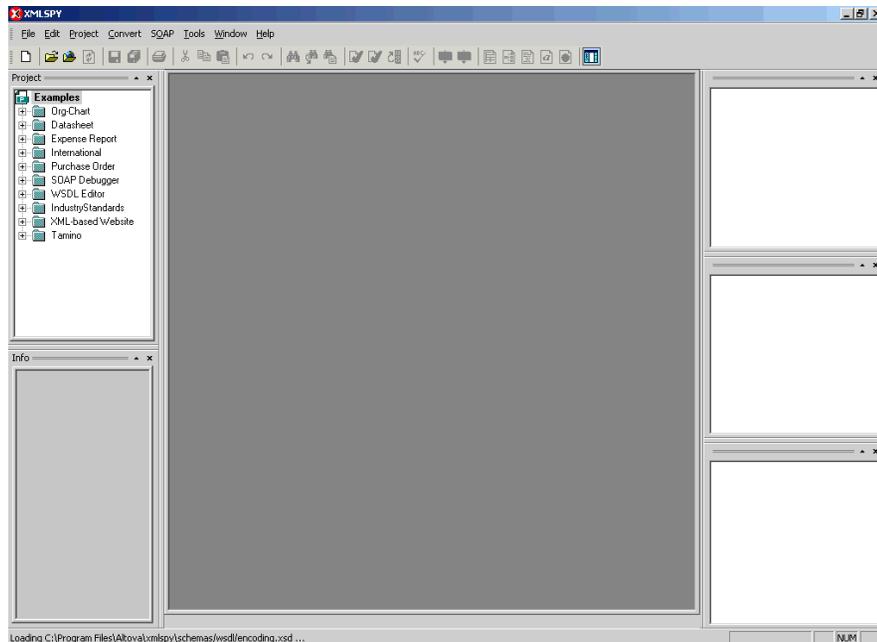
After you migrate your repository, you can export or import targets with their connection information and persistent data between repositories.

Procedure: How to Export a Target

To export a target:

1. Copy the iBSE administrative services for Application Explorer URL, for example:
<http://localhost:7777/ibse/IBSEServlet/admin/iwae.ibs?wsdl>
2. Open a third party XML editor, for example, XMLSPY.

The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.



3. From the SOAP menu, select *Create new SOAP request*.
The WSDL file location dialog box opens.
4. In the Choose a file field, paste the iBSE administrative services for Application Explorer URL.
5. Click *OK*.
The soap operation name dialog box opens and lists the available control methods.
6. Select the *EXPORTTARGET(EXPORTTARGET parameters)* control method and click *OK*.
A window opens that shows the structure of the SOAP envelope.
7. Locate the *Text view* icon in the tool bar.
In the following image, the pointer points to the Text view icon.



8. To display the structure of the SOAP envelope as text, click the *Text view* icon.

The <SOAP-ENV:Header> tag is not required and can be deleted from the SOAP envelope.

9. Locate the following section:

```
<m:EXPORTTARGET
xmlns:m="urn:schemas-iwaysoftware-com:dec2002:iwse:af">
<m:target>String</m:target>
<m:name>String</m:name>
</m:EXPORTTARGET>
```

- a. For the <m:target> tag, replace the String placeholder with the EIS target system name as it appears in Application Explorer and verify whether this value is case sensitive.
 - b. For the <m:name> tag, replace the String placeholder with the name of the target you want to export.
10. From the SOAP menu, select *Send request to server*.

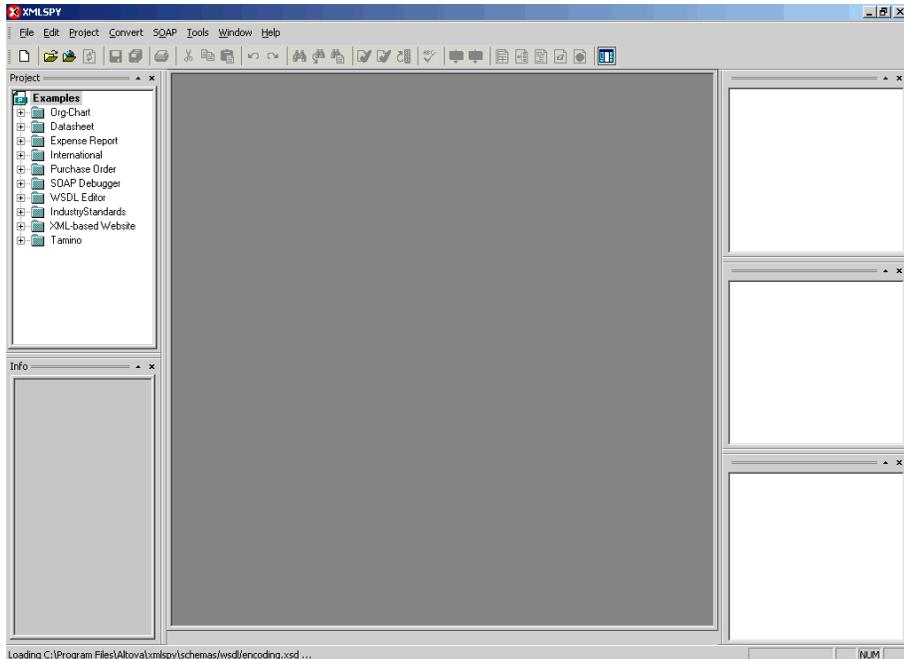
A response is returned that contains the <m: exporttime> and <m: contents> elements. You must use these elements when importing your target.

Procedure: How to Import a Target

To import a target:

1. Copy the iBSE administrative services for Application Explorer URL, for example:
<http://localhost:7777/ibse/IBSEServlet/admin/iwae.ibs?wsdl>
2. Open a third party XML editor, for example, XMLSPY.

The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.



3. From the SOAP menu, select *Create new SOAP request*.

The WSDL file location dialog box opens.

4. In the Choose a file field, paste the iBSE administrative services for Application Explorer URL and click *OK*.

The soap operation name dialog box opens and lists the available control methods.

5. Select the *IMPORTTARGET(IMPORTTARGET parameters)* control method and click *OK*.

A window opens, which shows the structure of the SOAP envelope.

6. Locate the *Text view* icon in the toolbar.

In the following image, the pointer points to the Text view icon.



7. To display the structure of the SOAP envelope as text, click the *Text view* icon.

The <SOAP-ENV:Header> tag is not required and can be deleted from the SOAP envelope.

8. Locate the following section:

```
<m:IMPORTTARGET
xmlns:m="urn:schemas-iwaysoftware-com:dec2002:iwse:af">
<m:targetinstance>
<m:target>String</m:target>
<m:name>String</m:name>
<m:description>String</m:description>
<m:repositoryid>String</m:repositoryid>
<m:exporttime>2001-12-17T09:30:47-05:00</m:exporttime>
<m:contents>R01GODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</m:contents>
</m:targetinstance>
</m:IMPORTTARGET>
```

- a. For the <m:target> tag, replace the String placeholder with the EIS target system name.
 - b. For the <m:name> tag, replace the String placeholder with the new name of the target you want to import.
 - c. For the <m:description> tag, replace the String placeholder with a description of the target.
 - d. For the <m:repositoryid> tag, copy and paste the contents of the <m:repositoryid> tag that was returned when you exported your target.
 - e. For the <m:exporttime> tag, copy and paste the contents of the <m:exporttime> tag that was returned when you exported your target.
 - f. For the <m:contents> tag, copy and paste the contents of the <m:contents> tag that was returned when you exported your target.
- 9.** From the SOAP menu, select *Send request to server*.

Retrieving or Updating Web Service Method Connection Information

After you migrate your repository, you can retrieve or update connection information for your Web service methods.

Procedure: How to Retrieve Web Service Method Connection Information

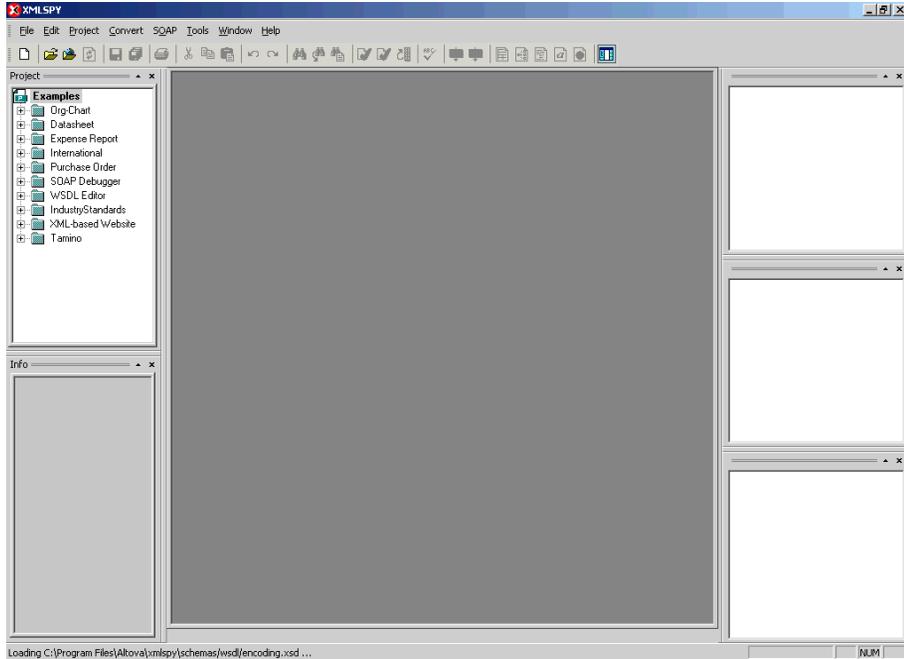
To retrieve Web service method connection information:

1. Copy the iBSE configuration service URL, for example:

```
http://localhost:7777/ibse/IBSEServlet/admin/iwconfig.ibs?wsdl
```

2. Open a third party XML editor, for example, XMLSPY.

The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.



3. From the SOAP menu, select *Create new SOAP request*.
The WSDL file location dialog box opens.
4. In the Choose a file field, paste the iBSE configuration service URL, and click *OK*.
The soap operation name dialog box opens and lists the available control methods.
5. Select the *GETMTHCONNECTION(GETMTHCONNECTION parameters)* control method and click *OK*.
A window opens, which shows the structure of the SOAP envelope.
6. Locate the *Text view* icon in the toolbar.
In the following image, the pointer points to the Text view icon.



7. To display the structure of the SOAP envelope as text, click the *Text view* icon.

Retrieving or Updating Web Service Method Connection Information

The <SOAP-ENV:Header> tag is not required and can be deleted from the SOAP envelope.

8. Locate the following section:

```
<m:GETMTHCONNECTION
xmlns:m="urn:schemas-iwaysoftware-com:jul2003:ibse:config">
<m:serviceName>String</m:serviceName>
<m:methodName>String</m:methodName>
</m:GETMTHCONNECTION>
```

- a. For the <m:serviceName> tag, replace the String placeholder with the name of the Web service.
 - b. For the <m:methodName> tag, replace the String placeholder with name of the Web service method.
9. From the SOAP menu, select *Send request to server*.

A response is returned that contains the <m:descriptor> element. You must use this element when updating your Web service method.

Procedure: How to Update Web Service Method Connection Information

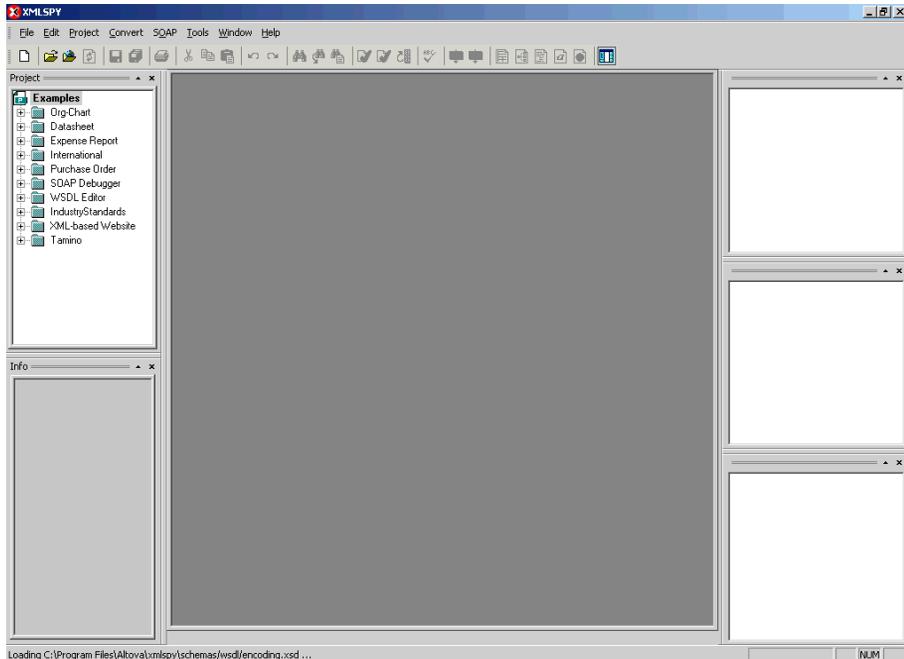
To update Web service method connection information:

1. Copy the iBSE configuration service URL, for example:

```
http://localhost:7777/ibse/IBSEServlet/admin/iwconfig.ibs?wsdl
```

2. Open a third party XML editor, for example, XMLSPY.

The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.



3. From the SOAP menu, select *Create new SOAP request*.
The WSDL file location dialog box opens.
4. In the Choose a file field, paste the iBSE configuration service URL, and click *OK*.
The soap operation name dialog box opens and lists the available control methods.
5. Select the *SETMTHCONNECTION(SETMTHCONNECTION parameters)* control method and click *OK*.
A window opens that shows the structure of the SOAP envelope.
6. Locate the *Text view* icon in the toolbar.
In the following image, the pointer points to the Text view icon.



7. To display the structure of the SOAP envelope as text, click the *Text view* icon.

Starting or Stopping a Channel Programmatically

The <SOAP-ENV:Header> tag is not required and can be deleted from the SOAP envelope.

8. Locate the following section:

```
<m:SETMTHCONNECTION
xmlns:m="urn:schemas-iwaysoftware-com:jul2003:ibse:config">
<m:servicename>String</m:servicename>
<m:methodname>String</m:methodname>
<m:descriptor format="" channel="">
  <m:option title="">
    <m:group title="">
      <m:param/>
    </m:group>
  </m:option>
</m:descriptor>
</m:SETMTHCONNECTION>
```

- a. For the <m:servicename> tag, replace the String placeholder with the name of the Web service.
 - b. For the <m:methodname> tag, replace the String placeholder with the name of the Web service method.
 - c. For the <m: descriptor> tag, copy and paste the contents of the <m: descriptor> tag that was returned when you retrieved Web Service method connection information.
- 9.** Modify the contents of the <m: descriptor> tag to change the existing Web Service method connection information.
- 10.** From the SOAP menu, select *Send request to server*.

Starting or Stopping a Channel Programmatically

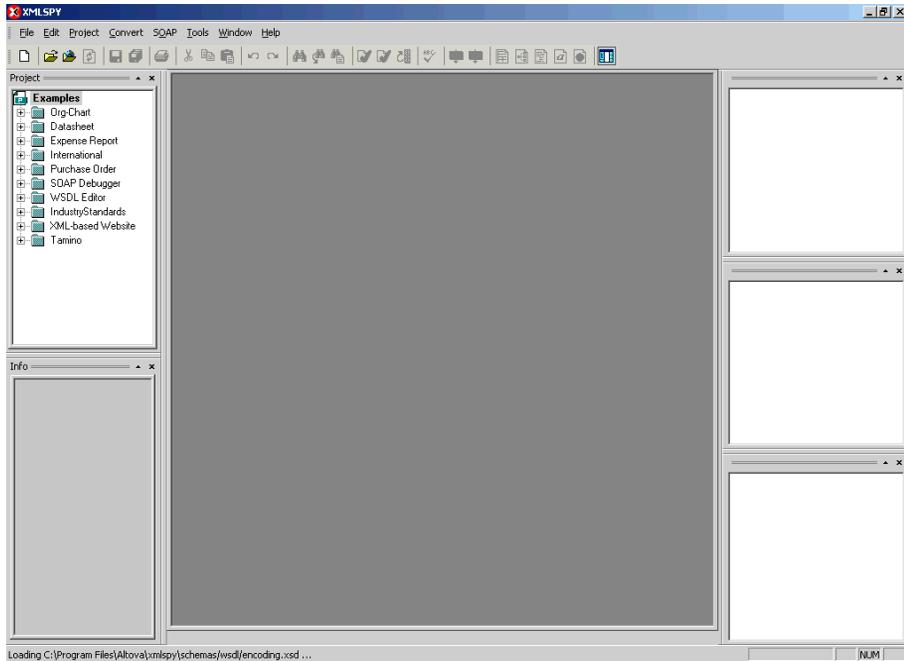
The following topic describes how to start or stop a channel programmatically.

Procedure: How to Start a Channel Programmatically

To start a channel programmatically:

1. Copy the iBSE control event URL, for example:
<http://localhost:7777/ibse/IBSEServlet/admin/iwevent.ibs?wsdl>
2. Open a third party XML editor, for example, XMLSPY.

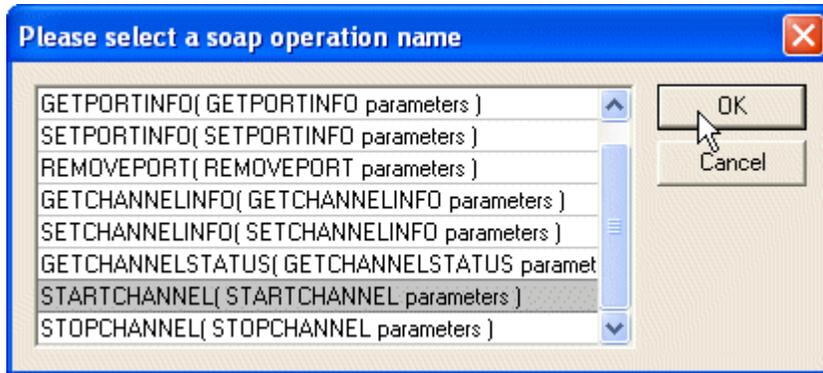
The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.



3. From the SOAP menu, select *Create new SOAP request*.
The WSDL file location dialog box opens.
4. In the Choose a file field, paste the iBSE control event URL, and click *OK*.

Starting or Stopping a Channel Programmatically

The following image shows the soap operation name dialog box that opens with a list of available control methods. You can select one and click OK or to escape from the dialog box, you can click Cancel.



5. Select the *STARTCHANNEL(STARTCHANNEL parameters)* control method and click *OK*.

A window opens, which shows the structure of the SOAP envelope.

6. Locate the *Text view* icon in the toolbar.

In the following image, the pointer points to the *Text view* icon.



7. To display the structure of the SOAP envelope as text, click the *Text view* icon.

The `<SOAP-ENV:Header>` tag is not required and can be deleted from the SOAP envelope.

8. Locate the following section:

```
<SOAP-ENV:Body>
  <m:STARTCHANNEL
    xmlns:m="urn:schemas-iwaysoftware-com:dec2002:iwse:event">
    <m:channel>String</m:channel>
  </m:STARTCHANNEL>
</SOAP-ENV:Body>
```

9. For the `<m:channel>` tag, replace the String placeholder with the name of the Channel you want to start.

10. From the SOAP menu, select *Send request to server*.

Procedure: How to Stop a Channel Programmatically

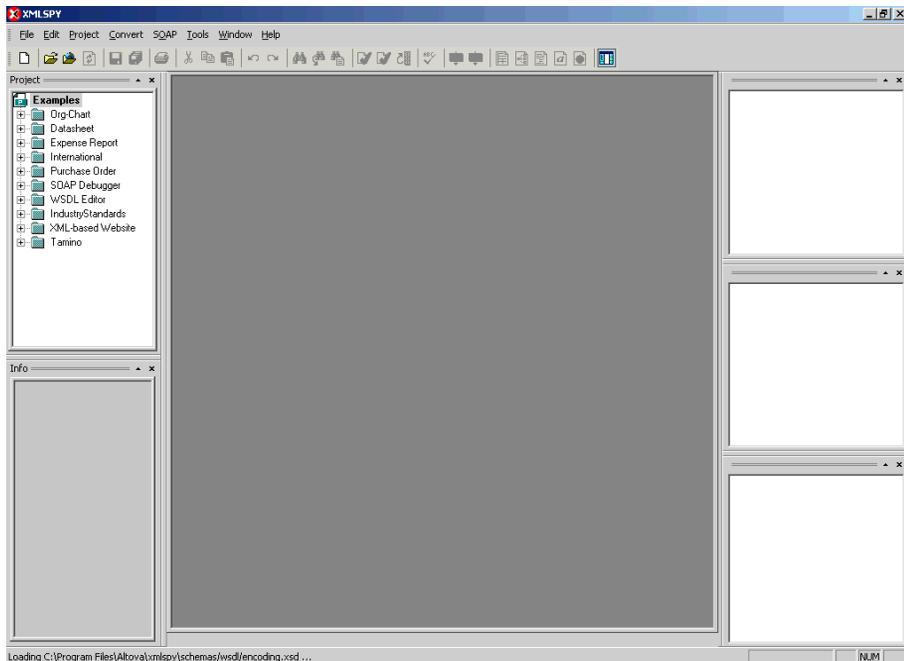
To stop a channel programmatically:

1. Copy the iBSE control event URL, for example:

<http://localhost:7777/ibse/IBSEServlet/admin/iwevent.ibs?wsdl>

2. Open a third party XML editor, for example, XMLSPY.

The following image shows the XMLSPY window. The upper left has a Project pane that contains a tree of sample files, and the lower left has a blank Info pane. The middle pane is blank. The right side is divided into three blank panes.



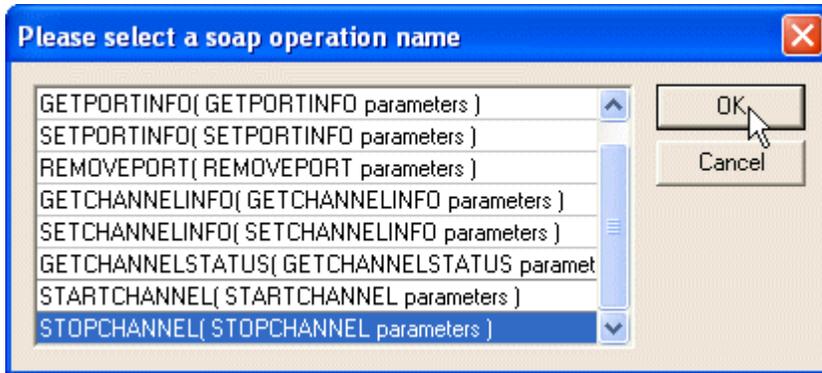
3. From the SOAP menu, select *Create new SOAP request*.

The WSDL file location dialog box opens.

4. In the Choose a file field, paste the iBSE control event URL, and click *OK*.

Starting or Stopping a Channel Programmatically

The following image shows the soap operation name dialog box that opens with a list of available control methods. You can select one and click OK or to escape from the dialog box, you can click Cancel.



5. Select the *STOPCHANNEL(STOPCHANNEL parameters)* control method and click *OK*.

A window opens, which shows the structure of the SOAP envelope.

6. Locate the *Text view* icon in the toolbar.

In the following image, the pointer points to the *Text view* icon.



7. To display the structure of the SOAP envelope as text, click the *Text view* icon.

The `<SOAP-ENV:Header>` tag is not required and can be deleted from the SOAP envelope.

8. Locate the following section:

```
<SOAP-ENV:Body>
  <m:STOPCHANNEL
    xmlns:m="urn:schemas-iwaysoftware-com:dec2002:iwse:event">
    <m:channel>String</m:channel>
  </m:STOPCHANNEL>
</SOAP-ENV:Body>
```

9. For the `<m:channel>` tag, replace the String placeholder with the name of the Channel you want to stop.

10. From the SOAP menu, select *Send request to server*.

CHAPTER 6

Troubleshooting

- Troubleshooting
- Integration Business Services Engine Error Messages

This section explains the limitations and workarounds when connecting to Oracle E-Business Suite.

The adapter-specific errors listed in this appendix can arise whether using the adapter with a JCA or with an iBSE configuration.

Troubleshooting

This topic provides troubleshooting information for BEA WebLogic Adapter for Oracle E-Business Suite.

Reference: Application Explorer

Error	Solution
Cannot connect to the BEA WebLogic Adapter for Oracle E-Business Suite from Application Explorer.	Ensure that: <ul style="list-style-type: none"> • Oracle E-Business Suite is running. • The Oracle E-Business Suite user ID and password are correct. • The port number is correct.
The following error message appears: (login error)	You have provided invalid connection information for Oracle E-Business Suite or the wrong JAR file is in the lib directory. See the <i>BEA WebLogic ERP Adapter Installation and Configuration</i> manual for information on JAR files.
Oracle Applications does not appear in the Application Explorer Adapter node list.	Ensure that the Oracle Applications JAR files are added to the lib directory. See the <i>BEA WebLogic ERP Adapter Installation and Configuration</i> manual for information on JAR files.

Reference: JCA

Error	Solution
In Application Explorer, the following error message appears when you attempt to connect to a JCA configuration: Could not initialize JCA	In the Details tab in the right pane, ensure that the directory specified in the Home field points to the correct directory, for example, C:\Program Files\iWay55.

Integration Business Services Engine Error Messages

This topic discusses the different types of errors that can occur when processing Integration Business Services through Integration Business Services Engine (iBSE).

General Error Handling in iBSE

iBSE serves as both a SOAP gateway into the adapter framework and as the engine for some of the adapters. In both design time and execution time, various conditions can cause errors in iBSE when Web services that use adapters are running. Some of these conditions and resulting errors are exposed the same way, regardless of the specific adapter; others are exposed differently, based on the adapter being used. This topic explains what you can expect when you encounter some of the more common error conditions on an adapter-specific basis.

Usually, the SOAP gateway (agent) inside iBSE passes a SOAP request message to the adapter required for the Web service. If an error occurs, how it is exposed depends on the adapter and the API or interfaces that the adapter uses. A few scenarios cause the SOAP gateway to generate a SOAP fault. In general, anytime the SOAP agent inside iBSE receives an invalid SOAP request, a SOAP fault element is generated in the SOAP response. The SOAP fault element contains fault string and fault code elements. The fault code contains a description of the SOAP agent error.

The following SOAP response document results when iBSE receives an invalid SOAP request:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Client</faultcode>
      <faultstring>Parameter node is missing</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In this example, iBSE did not receive an element in the SOAP request message that is mandatory for the WSDL for this Web service.

Adapter-Specific Error Handling

When an adapter raises an exception during execution, the SOAP agent in iBSE produces a SOAP fault element in the generated SOAP response. The SOAP fault element contains fault code and fault string elements. The fault string contains the native error description from the adapter target system. Since adapters use the target system interfaces and APIs, whether or not an exception is raised depends on how the target systems interface or API treats the error condition. If a SOAP request message is passed to an adapter by the SOAP agent in iBSE, and that request is invalid based on the WSDL for that service, the adapter may raise an exception yielding a SOAP fault.

While it is almost impossible to anticipate every error condition that an adapter may encounter, the following is a description of how adapters handle common error conditions and how they are then exposed to the Web services consumer application.

BEA WebLogic Adapter for Oracle E-Business Suite Invalid SOAP Request

When the Oracle agent receives a SOAP request message that does not conform to the WSDL for the Web service being executed, the following SOAP response is generated:

```
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <m:CARRIERResponse xmlns:m="urn:schemas-iwaysoftware-com:iwse"
xmlns="urn:schemas-iwaysoftware-com:iwse"
cid="2A3CB42703EB20203F91951B89F3C5AF">
      <PS8>
        <error>Cannot find Component Interface {VARRIER}
(91,2) Initialization
failed (90,7)Not Authorized (90,6)Failed to execute PSSession request
Cannot find Component Interface {VARRIER} (91,2)</error>
      </PS8>
    </m:CARRIERResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Failure to Connect to Oracle Applications

When the BEA WebLogic Adapter for Oracle E-Business Suite cannot connect to Oracle, the following SOAP response is generated.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
  <SOAP-ENV:Fault>
    <faultcode>SOAP-ENV:Server</faultcode>
    <faultstring>java.lang.Exception: Error Logon to Oracle Applications
      System</faultstring>
  </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Invalid SOAP Request

When the adapter receives a SOAP request message that does not conform to the WSDL for the Integration Business Services being executed, the following SOAP response is generated.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
  <SOAP-ENV:Fault>
    <faultcode>SOAP-ENV:Server</faultcode>
    <faultstring>RPC server connection failed: Connection refused:
connect</faultstring>
  </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Empty Result From an Adapter Request

When the adapter executes a SOAP Request using input parameters passed that do not match records in the target system, the following SOAP response is generated.

Note: The condition for this adapter does not yield a SOAP fault.

```
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <m:RunDBQueryResponse xmlns:m="urn:schemas-iwaysoftware-com:iwse"
xmlns="urn:schemas-iwaysoftware-com:iwse"
cid="2A3CB42703EB20203F91951B89F3C5AF">
      <RunDBQueryResult run="1" />
    </m:RunDBQueryResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

APPENDIX A

Using Application Explorer in BEA WebLogic Workshop to Create XML Schemas and Web Services

Topics:

- Starting Application Explorer in BEA WebLogic Workshop
- Creating a New Configuration
- Connecting to Oracle E-Business Suite
- Creating an XML Schema
- Creating an Integration Business Service
- Adding a Control for a Resource in BEA WebLogic Workshop
- Adding an Extensible CCI Control to a BEA WebLogic Workshop Application

This section describes how to use Java Swing Application Explorer running in BEA WebLogic Workshop to create XML schemas and Web services for Oracle E-Business Suite.

Starting Application Explorer in BEA WebLogic Workshop

The server must be started on which Application Explorer is running. Before you can use Application Explorer, you must start BEA WebLogic Server.

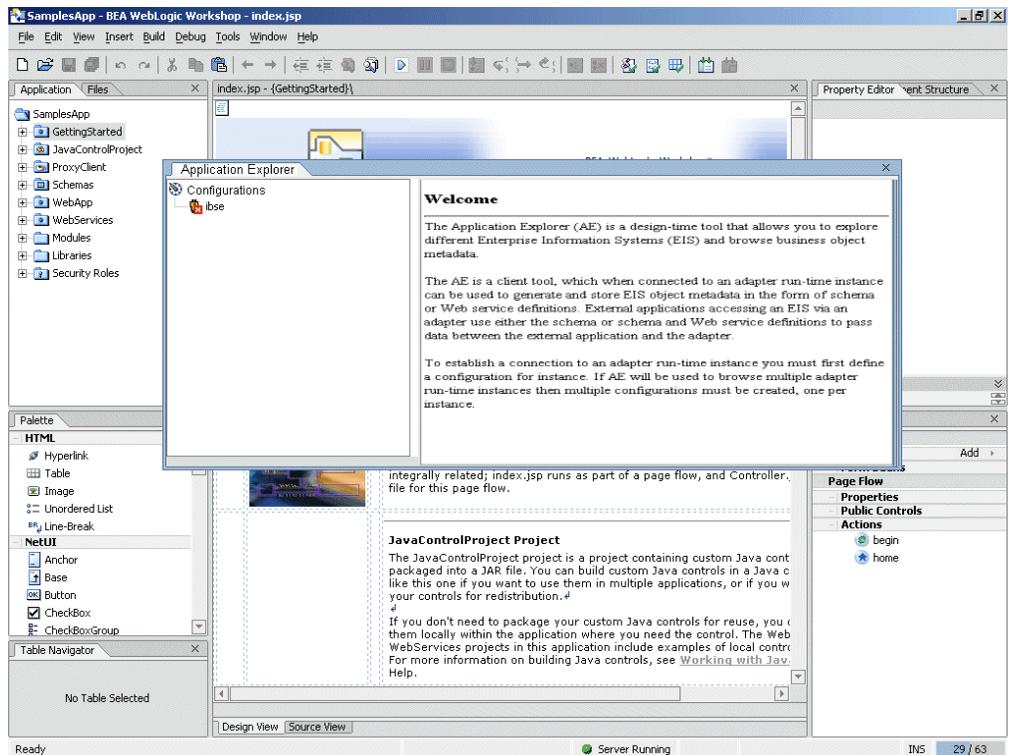
You can run Application Explorer in BEA WebLogic Workshop using an Integration Business Services Engine (IBSE) configuration or J2EE Connector Architecture (JCA) configuration.

Procedure: How to Start Application Explorer in BEA WebLogic Workshop

To start Application Explorer in BEA WebLogic Workshop:

1. Ensure that BEA WebLogic Server is running.
2. Start BEA WebLogic Workshop.
3. From the BEA WebLogic Workshop View menu, select *Windows* and then, *Application Explorer*.

Application Explorer opens in BEA WebLogic Workshop, as shown in the following image. This image also shows the Application Explorer welcome statement that appears on the right when you select the Configurations node on the left.



You can resize and drag-and-drop the Application Explorer window within BEA WebLogic Workshop. For example, you can drag it to the upper part of BEA WebLogic Workshop.

Creating a New Configuration

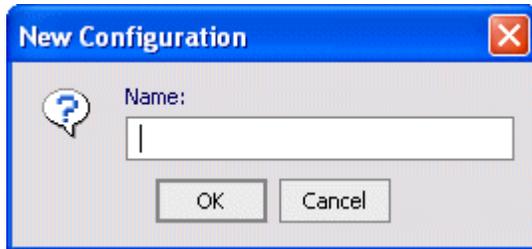
Before you can start using Application Explorer, you must define a new configuration for iBSE or JCA.

Procedure: How to Create a New Configuration for iBSE or JCA

To create a new configuration:

1. Right-click *Configurations* and select *New*.

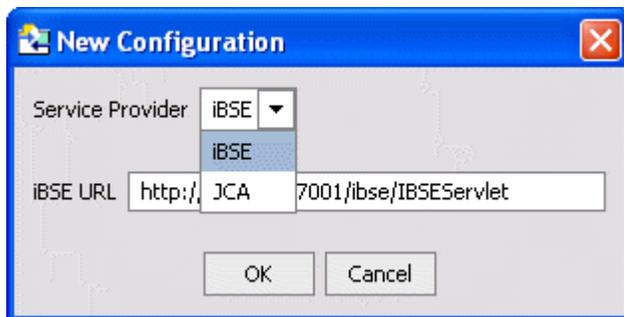
The New Configuration dialog box opens, as shown in the following image.



2. Type a name for the new configuration and click *OK*.

Note: If you are creating a new JCA configuration, type *base* in the Name field. You must use this value if you are pointing to the default configuration.

A dialog box opens, as shown in the following image.



3. From the Service Provider drop-down list, select *iBSE* or *JCA*.

If you select iBSE, type the URL for iBSE, for example,

<http://localhost:7001/ibse/IBSEServlet>

where:

`localhost`

Is where your application server is running.

If you select JCA, enter the full path to the directory where iWay 5.5 is installed, for example,

`C:\Program Files\iWay55`

where:

`iWay55`

Is the full path to your iWay installation.

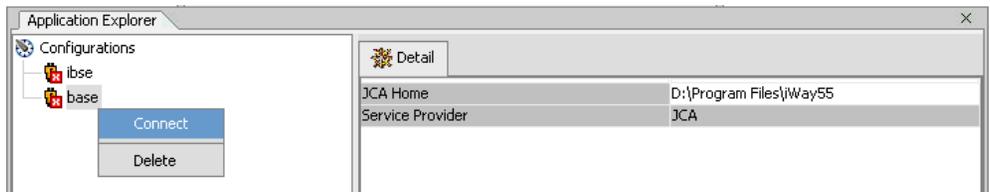
4. Click OK.

A node representing the new iBSE or JCA instance appears below the root Configurations node. When you select a node under Configurations, the right pane provides details about that node.

After you add your configuration, you must connect to it.

5. Right-click the configuration to which you want to connect and select *Connect*.

The following image shows a selected node with the right-click menu on the left and details about the node on the right.



When you connect to base, the Service Adapters and Event Adapters nodes appear, as shown in the following image.



When you connect to iBSE, the Service Adapters, Event Adapters, and Integration Business Services nodes appear, as shown in the following image.



6. To display the service and event adapters that are installed, expand each node.

The Service Adapters list includes an Oracle Applications node that enables you to connect to Oracle metadata and create XML request and response schemas to use to listen for events or create Web services. For more information, see *Creating an Integration Business Service* on page A-17.

The Event Adapters list includes an Oracle Applications node that enables you to create ports and channels for Oracle event handling. For more information, see Appendix B, *Using Application Explorer in BEA WebLogic Workshop for Event Handling*.

Connecting to Oracle E-Business Suite

To browse Oracle E-Business Suite tables, you must create an Oracle E-Business Suite target and connect to it. The target serves as your connection point. You must establish a connection to Oracle every time you start Application Explorer or after you disconnect from Oracle.

The left pane displays the application systems supported by Application Explorer. These are based on the adapters you installed and are licensed to use.

Creating and Connecting to a Target

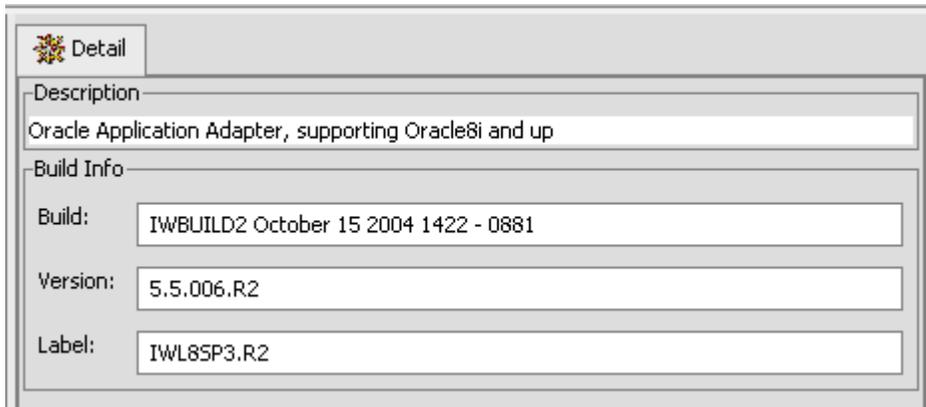
To connect to Oracle E-Business Suite for the first time, you must create a new target. The target is automatically saved after it is created.

Procedure: How to Create a New Target

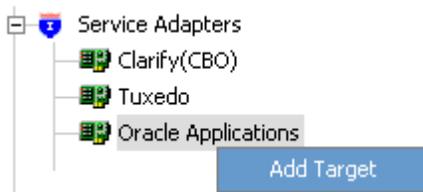
To create a new target:

1. In the left pane, expand the *Service Adapters* node and click the *Oracle Applications* node.

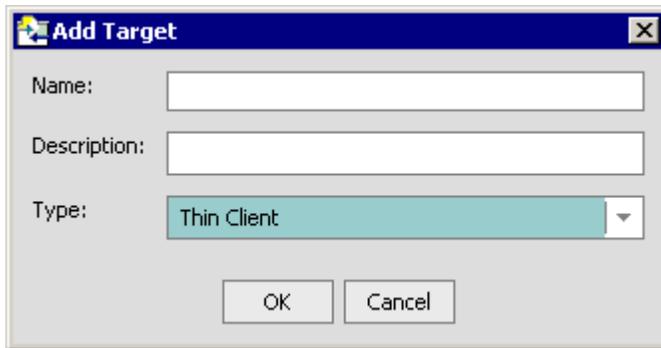
Descriptive information (for example, label and product version) for the adapter appears on the right. The following image is an example of the details for the Oracle Applications node.



2. Right-click the *Oracle Applications* node and select *Add Target*.



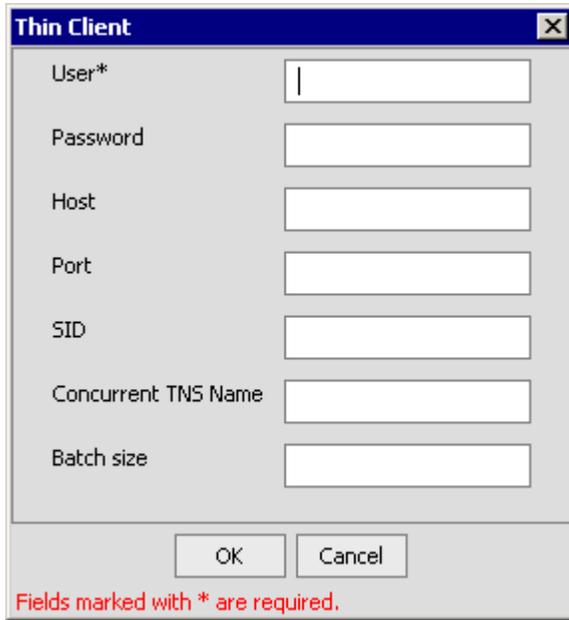
The Add Target dialog box opens, as shown in the following image. The dialog box provides three fields to define the target and two action buttons.



- a. In the Name field, type a descriptive name for the target, for example, OracleTarget.
 - b. In the Description field, type a brief description of the target.
 - c. From the Type drop-down list, select the type of Oracle client to which you want to connect. The options are:
 - Thin Client
 - OCI Client
- 3.** Click OK.

The dialog box that appears depends on the target type selection.

If you selected a Thin Client target type, the Thin Client window opens, as shown in the following image. The Thin Client window provides fields to specify connection information for Oracle and the application server that is hosting Oracle, and two action buttons.



The image shows a 'Thin Client' dialog box with the following fields and controls:

- User***: A text input field with a red asterisk indicating it is required.
- Password**: A text input field.
- Host**: A text input field.
- Port**: A text input field.
- SID**: A text input field.
- Concurrent TNS Name**: A text input field.
- Batch size**: A text input field.
- Buttons**: 'OK' and 'Cancel' buttons at the bottom.
- Note**: A red text note at the bottom states 'Fields marked with * are required.'

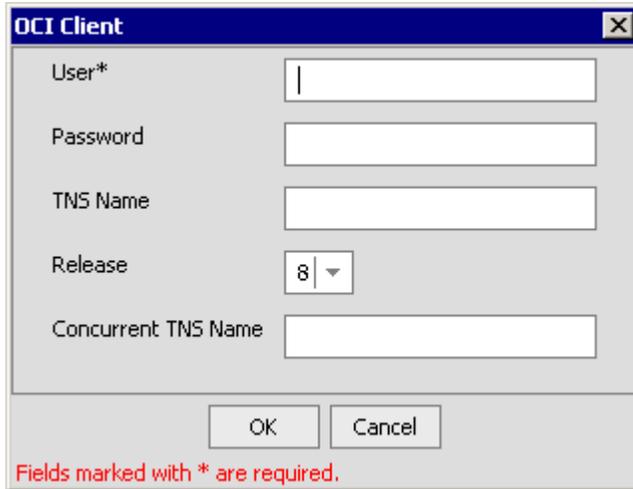
Note: The Oracle E-Business Suite connection parameters are consistent with those found in your Oracle E-Business Suite system. For more information on parameter values that are specific to your Oracle E-Business Suite configuration, consult your Oracle E-Business Suite system administrator.

The following table lists and describes each connection parameter for a Thin Client target.

Field	Description
User	The Oracle database user ID to access the Oracle database underlying the Oracle E-Business Suite system. The user ID must have database access to the interface tables being accessed.
Password	The password associated with the specified user ID.
Host	The name of the server on which the Oracle E-Business Suite database instance resides.
Port	The port number on which the database is listening.

Field	Description
SID	A unique name of the database service, chosen by the database administrator or the person who installed Oracle E-Business Suite.
Concurrent TNS Name	<p>The logical name of the Oracle E-Business Suite database instance. (You can obtain this from the database administrator.)</p> <p>The TNS name that is found in the tnsnames.ora file that contains an entry for that database and that resides on the Oracle E-Business Suite database server. This is <i>not</i> the TNS name that is registered on the server on which the BEA WebLogic Adapter for Oracle E-Business Suite is running (unless the adapter and the Oracle database reside on the same server).</p> <p>If a TNS name does not exist, you must create it. For more information about creating a TNS name, see your Oracle administrator.</p>
Batch size	<p>The number of request document records that the adapter buffers before inserting them into an Oracle E-Business Suite table. When the adapter reaches the end of the request document, it inserts any remaining buffered records.</p> <p>The adapter tracks this batch limit separately for each table, not aggregately for all tables. For example, if you set the batch size property to 25, and the adapter accumulates 18 input records for the GL_INTERFACE table and 25 for the BUDGET_INTERFACE table, the adapter continues to hold records for GL_INTERFACE but inserts the records for BUDGET_INTERFACE.</p> <p>The batch size property enables you to optimize I/O: inserting too few records at a time increases I/O overhead, and inserting too many at a time ties up database resources.</p>

If you selected an OCI Client target type, the OCI Client window opens, as shown in the following image. The Thin Client window provides connection parameter fields and two action buttons.



Note: The Oracle E-Business Suite connection parameters are consistent with those found in your Oracle E-Business Suite system. For more information on parameter values that are specific to your Oracle E-Business Suite configuration, consult your Oracle E-Business Suite system administrator.

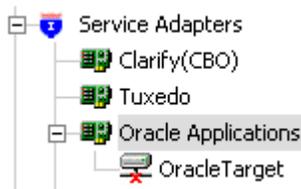
The following table lists and describes each connection parameter for an OCI Client target.

Field	Description
User	The Oracle database user ID to access the Oracle database underlying the Oracle E-Business Suite system. The user ID must have database access to the interface tables being accessed.
Password	The password associated with the specified user ID.
TNS Name	The logical name of the Oracle E-Business Suite database instance. (You can obtain this from the database administrator.) The TNS name that is found in the tnsnames.ora file that contains an entry for that database and that <i>resides on the server running the adapter</i> . If a TNS name does not exist, you must create it. For information about creating a TNS name, see your Oracle administrator.

Field	Description
Release	The release number of the Oracle E-Business Suite instance. Select it from the list of releases supported by the adapter.
Concurrent TNS Name	The logical name of the Oracle E-Business Suite database instance. (You can obtain this from the database administrator.) The TNS name that is found in the tnsnames.ora file that contains an entry for that database and that <i>resides on the server hosting the Oracle E-Business Suite database</i> . If a TNS name does not exist, you must create it. For information about creating a TNS name, see your Oracle administrator.

4. Enter values for the connection parameters. The Thin Client and OCI Client connection parameter values are defined in the tables found in Step 3.
5. Click *OK*.

In the left pane, the new target (OracleTarget) appears below the Oracle Applications node, as shown in the following image.



You can now connect to the target you defined.

Procedure: How to Connect to a Target

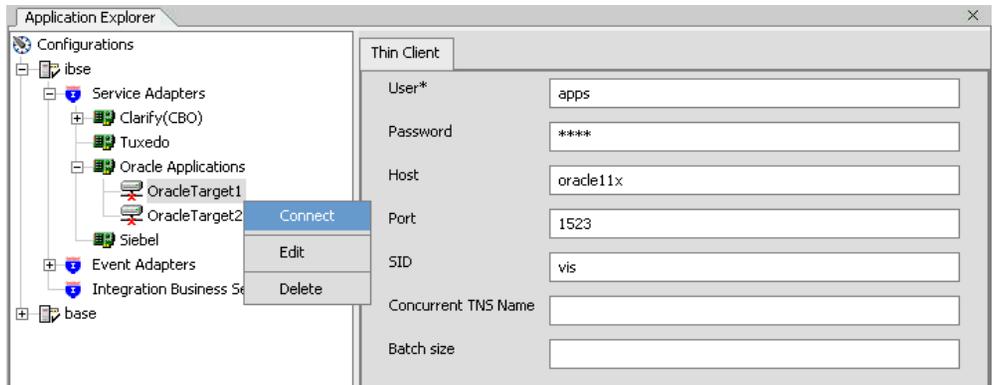
To connect to an Oracle Applications target:

1. In the left pane, expand the *Oracle Applications* node and select the target to which you want to connect.

The connection parameters for the target, which were defined when the target was created, open on the right. If you select a Thin Client target to connect to, the Thin Client connection parameters appear. If you select an OCI Client target, the OCI Client connection parameters appear.

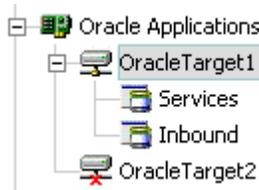
2. In the Password field, type a valid password.

The following image shows an example of a Thin Client target type, *OracleTarget1*.



3. In the left pane, right-click the target node you want to open and select *Connect*.

The target node in the left pane changes appearance to reflect that a connection is made, as shown in the following image. OracleTarget1 is connected, while OracleTarget2 is disconnected.



Managing a Target

Although you can maintain multiple open connections to different application systems, we recommend that you close connections when they are not in use. After you disconnect, you can modify an existing target.

You can modify the connection parameters when your system properties change. You also can delete a target. The following procedures describe how to disconnect from a target, edit a target, and delete a target.

Procedure: How to Disconnect From a Target

To disconnect from a target:

1. Right-click the Oracle target from which you want to disconnect. The following image is an example of a target and its right-click menu.



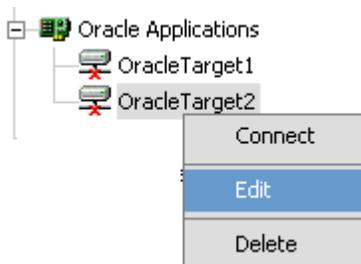
2. Select *Disconnect*.

Disconnecting from the application system drops the target, but the node remains. The OracleTarget node in the left pane changes to reflect that you disconnected from the target.

Procedure: How to Edit a Target

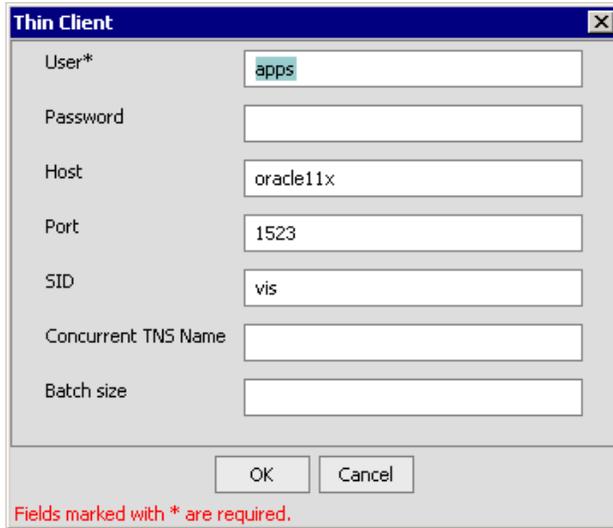
To edit a target:

1. Ensure that the target you want to edit is disconnected. The following image is an example of a target and its right-click menu.



2. In the left pane, right-click a disconnected target and select *Edit*.

A dialog box opens, displaying the target properties. The target type determines whether the Thin Client dialog box or the OCI Client dialog box opens. The following image is an example of a Thin Client dialog box.

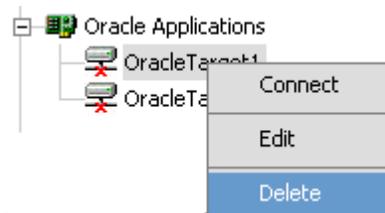


3. Change the target properties in the dialog box as required and click *OK*.

Procedure: How to Delete a Target

To delete a target:

1. Right-click the target. The following image is an example of a target and its right-click menu.



2. Select *Delete*.

The target node disappears from the left pane.

Creating an XML Schema

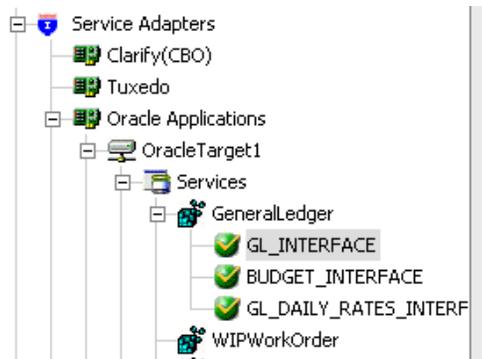
After you create a new configuration and connect to Oracle, Application Explorer enables you to create a request or response schema.

Procedure: How to Create Request and Response Schemas

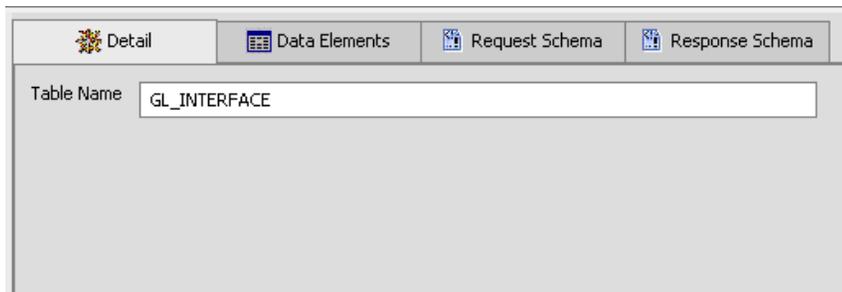
To create request and response schemas:

1. Connect to the desired target.
2. Under *Service Adapters*, expand the *Oracle Applications* node, the target node to which you are connected, and the *Services* node, and select the table for which you want to create the schema.

The following image shows an example of selecting the `GL_INTERFACE` table under the *Services* node.

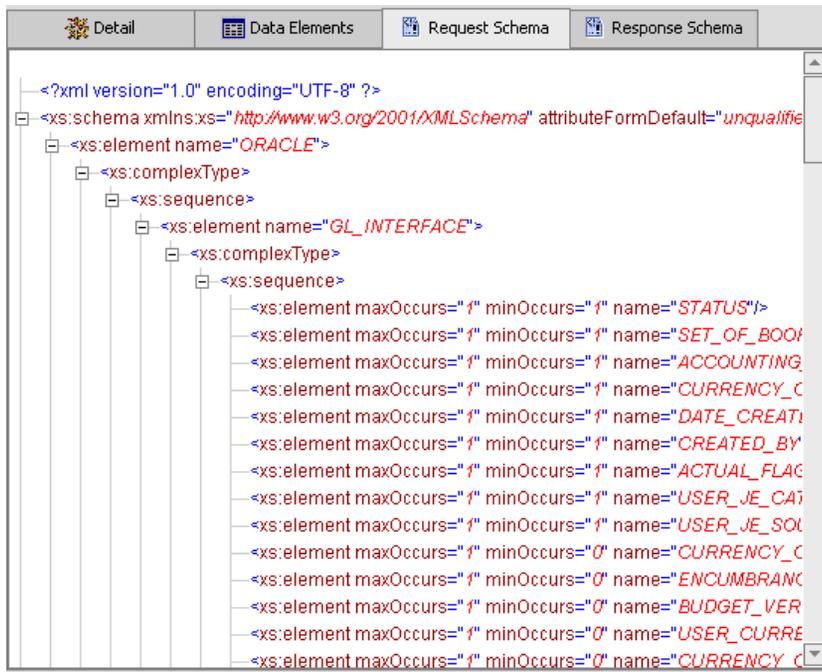


The Detail tab opens on the right, as shown in the following image. Three additional tabs (*Data Elements*, *Request Schema*, and *Response Schema*) are available from this display.



3. To generate and view schema for the table, select the *Request Schema* tab or the *Response Schema* tab.

The schema appears on the right. The following image shows an example of request schema.



Reference: Schema Location

Application Explorer stores the schemas it creates in subdirectories under the iWay home directory of the machine where it is installed. The exact location of the schemas differs depending on whether you deploy Application Explorer with an iBSE or a JCA configuration.

- **When using the adapter with an iBSE configuration**, the schemas are stored under a \schemas subdirectory of the iWay home directory, for example,

`C:\Program Files\iWay55\bea\ibse\wsdl\schemas\service\Oracle Applications\OracleTarget\SA45280C`

where:

`OracleTarget`

Is the name of the Oracle E-Business Suite target.

`SA45280C`

Is a randomly generated folder name that contains the schema files.

- **When using the adapter with a JCA configuration**, the schemas are stored under a \schemas subdirectory of the iWay home directory, for example,

`C:\Program Files\iWay55\config\base\schemas\Oracle Applications
\OracleTarget`

where:

`OracleTarget`

Is the name of the connection to the Oracle Applications system you defined in Application Explorer. Application Explorer stores the schemas in this directory.

Creating an Integration Business Service

You can create an integration business service (also known as a Web service) for an Oracle E-Business Suite interface table. To create a business service, you must deploy the BEA WebLogic Adapter for Oracle E-Business Suite in a business services environment using the Integration Business Services Engine (iBSE). iBSE exposes functionality as Web services and serves as a gateway to heterogeneous back-end applications and databases.

A Web service is a self-contained, modularized function that can be published and accessed across a network using open standards. It is the implementation of an interface by a component and is an executable entity. For the caller or sender, a Web service can be considered as a “black box” that may require input and delivers a result. Web services integrate within an enterprise as well as across enterprises on any communication technology stack, whether asynchronous or synchronous, in any format.

You can make a Web service available to other services within a host server by generating WSDL (Web Services Description Language) from the Web service.

Because Application Explorer runs within BEA WebLogic Workshop, you can easily incorporate Web services into BEA WebLogic Workflows. To enable BEA WebLogic Workshop to use Web services, you export the WSDL to a directory accessible to BEA WebLogic Workshop.

Ensure you properly configure the servlet iBSE. For more information on installing and deploying components, see the *BEA WebLogic ERP Adapter Installation and Configuration* manual.

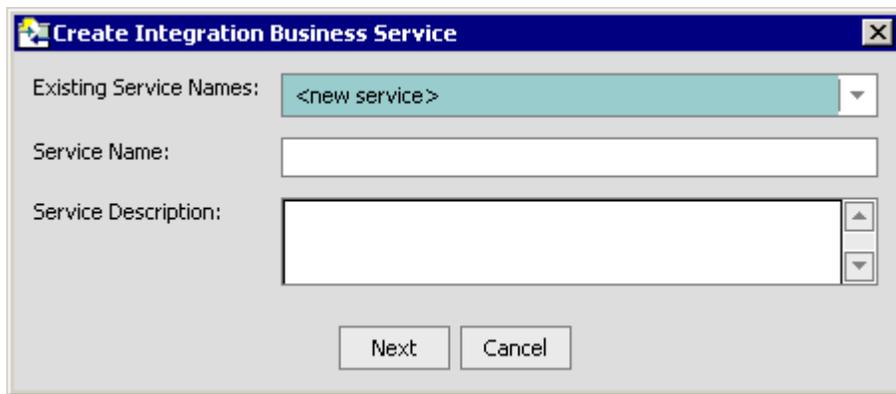
Note: In a J2EE Connector Architecture (JCA) implementation of adapters, Web services are not available. When the adapters are deployed to use the Connector for JCA, the Common Client Interface provides integration services using the adapters. For more information, see the *BEA WebLogic ERP Adapter Installation and Configuration* manual.

Procedure: How to Create an Integration Business Service

To create an integration business service for an Oracle E-Business Suite table:

1. Under the Service Adapters node, expand the *Oracle Applications* node, the target node to which you are connected, and the *Services* node, then select the service and interface table for which you want to create a business service.
2. Right-click the interface table and select *Create Integration Business Service*.

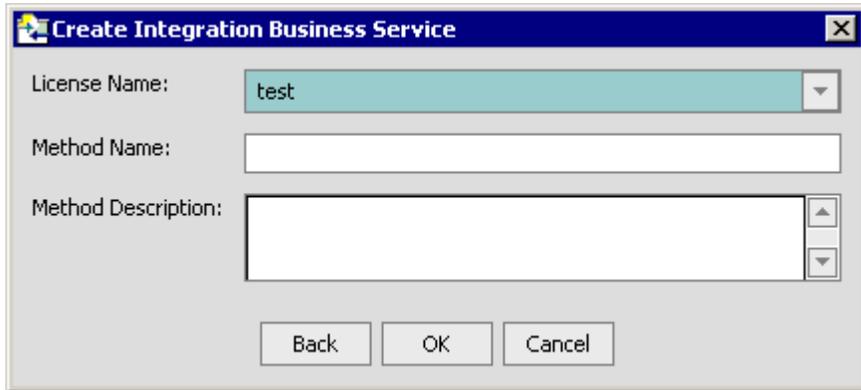
The Create Integration Business Service dialog box opens. This dialog box, as shown in the following image, provides three fields in which to define the service and two action buttons.



The image shows a Windows-style dialog box titled "Create Integration Business Service". It has a close button (X) in the top right corner. The dialog contains three input fields: "Existing Service Names" with a dropdown menu currently showing "<new service>", "Service Name" with an empty text box, and "Service Description" with an empty text box and vertical scroll arrows. At the bottom of the dialog are two buttons: "Next" and "Cancel".

- a. From the Existing Service Names drop-down list, select whether you want to create a new service (<new service>) or use an existing service.
 - b. If you selected a new service, type a name for the new business service in the Service Name field.
 - c. If you selected a new service, type a brief description of the new business service in the Service Description field.
3. Click *Next*.

The Create Integration Business Service dialog box displays additional fields to define a method, as shown in the following image.



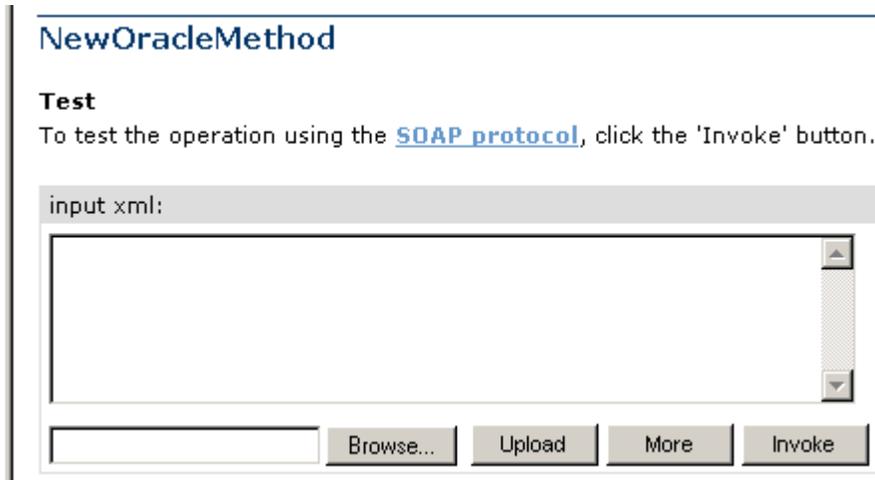
- a. From the License Name drop-down list, select a license.
 - b. In the Method Name field, type a name for the method.
 - c. In the Method Description field, type a brief description for the method.
4. Click **OK**.

On the left, the new business service and method appear under the Integration Business Services node. In the following image, the example service is NewOracleService, and the example method is NewOracleMethod.



5. Select the node for which you created the business service. On the right, two links enable you to view details of a service (Service Description) and test the service (link with the method name), as shown in the previous image.
6. To test the method, click the method link on the right.

The test window opens, as shown in the following image.



7. To invoke the service, enter a sample XML document in the input xml field.
For sample input XML, see *Testing an Integration Business Service on page A-21*.
8. Click *Invoke*.

The result appears in the right pane. An example of test results is shown in the following image.



Example: Testing an Integration Business Service

The following example is sample input XML for testing an Integration Business Service.

```

<ORACLE>
  <WIP_JOB_SCHEDULE_INTERFACE>
    <LAST_UPDATE_DATE>2002-12-02</LAST_UPDATE_DATE>
    <LAST_UPDATED_BY>1001611</LAST_UPDATED_BY>
    <CREATION_DATE>2002-12-02</CREATION_DATE>
    <CREATED_BY>1001611</CREATED_BY>
    <LAST_UPDATE_LOGIN/>
    <REQUEST_ID/>
    <PROGRAM_ID/>
    <PROGRAM_APPLICATION_ID/>
    <PROGRAM_UPDATE_DATE/>
    <GROUP_ID>2</GROUP_ID>
    <SOURCE_CODE/>
    <SOURCE_LINE_ID/>
    <PROCESS_TYPE/>
    <ORGANIZATION_ID/>
    <LOAD_TYPE>1</LOAD_TYPE>
    <STATUS_TYPE>3</STATUS_TYPE>
    <OLD_STATUS_TYPE/>
    <LAST_UNIT_COMPLETION_DATE/>
    <OLD_COMPLETION_DATE/>
    <PROCESSING_WORK_DAYS/>
    <DAILY_PRODUCTION_RATE/>
    <LINE_ID/>
    <PRIMARY_ITEM_ID>155</PRIMARY_ITEM_ID>
    <BOM_REFERENCE_ID/>
    <ROUTING_REFERENCE_ID/>
    <BOM_REVISION_DATE/>
    <ROUTING_REVISION_DATE/>
    <WIP_SUPPLY_TYPE>7</WIP_SUPPLY_TYPE>
    <CLASS_CODE>Discrete</CLASS_CODE>
    <LOT_NUMBER/>
    <LOT_CONTROL_CODE/>
    <JOB_NAME/>
    <DESCRIPTION/>
    <FIRM_PLANNED_FLAG/>
    <ALTERNATE_ROUTING_DESIGNATOR/>
    <ALTERNATE_BOM_DESIGNATOR> </ALTERNATE_BOM_DESIGNATOR>
    <START_QUANTITY>100</START_QUANTITY>
    <OLD_START_QUANTITY/>
    <WIP_ENTITY_ID/>
    <REPETITIVE_SCHEDULE_ID/>
    <ERROR/>
    <PARENT_GROUP_ID/>
    <ATTRIBUTE_CATEGORY/>

```

```
<ATTRIBUTE1 />
<ATTRIBUTE2 />
<ATTRIBUTE3 />
<ATTRIBUTE4 />
<ATTRIBUTE5 />
<ATTRIBUTE6 />
<ATTRIBUTE7 />
<ATTRIBUTE8 />
<ATTRIBUTE9 />
<ATTRIBUTE10 />
<ATTRIBUTE11 />
<ATTRIBUTE12 />
<ATTRIBUTE13 />
<ATTRIBUTE14 />
<ATTRIBUTE15 />
<INTERFACE_ID />
<LAST_UPDATED_BY_NAME />
<CREATED_BY_NAME />
<PROCESS_PHASE>2</PROCESS_PHASE>
<PROCESS_STATUS>1</PROCESS_STATUS>
<ORGANIZATION_CODE>M1</ORGANIZATION_CODE>
<FIRST_UNIT_START_DATE>2002-12-02</FIRST_UNIT_START_DATE>
<FIRST_UNIT_COMPLETION_DATE>2002-12-02</FIRST_UNIT_COMPLETION_DATE>
<LAST_UNIT_START_DATE>2002-12-02</LAST_UNIT_START_DATE>
<SCHEDULING_METHOD />
<LINE_CODE />
<PRIMARY_ITEM_SEGMENTS />
<BOM_REFERENCE_SEGMENTS />
<ROUTING_REFERENCE_SEGMENTS />
<ROUTING_REVISION />
<BOM_REVISION />
<COMPLETION_SUBINVENTORY />
<COMPLETION_LOCATOR_ID />
<COMPLETION_LOCATOR_SEGMENTS />
<SCHEDULE_GROUP_ID />
<SCHEDULE_GROUP_NAME />
<BUILD_SEQUENCE />
<PROJECT_ID />
<PROJECT_NAME />
<TASK_ID />
<TASK_NAME />
<NET_QUANTITY>100</NET_QUANTITY>
<DESCRIPTIVE_FLEX_SEGMENTS />
<PROJECT_NUMBER />
<TASK_NUMBER />
<PROJECT_COSTED />
<END_ITEM_UNIT_NUMBER />
<OVERCOMPLETION_TOLERANCE_TYPE />
```

```

<OVERCOMPLETION_TOLERANCE_VALUE/>
<KANBAN_CARD_ID/>
<PRIORITY>2</PRIORITY>
<DUE_DATE>2002-12-02</DUE_DATE>
<ALLOW_EXPLOSION>Y</ALLOW_EXPLOSION>
<HEADER_ID/>
<DELIVERY_ID/>
<COPRODUCTS_SUPPLY/>
<DUE_DATE_PENALTY/>
<DUE_DATE_TOLERANCE/>
</WIP_JOB_SCHEDULE_INTERFACE>
</ORACLE>

```

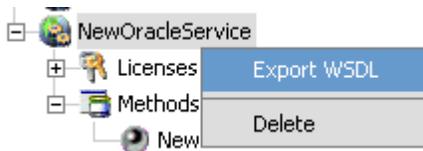
Exporting WSDL for Use in BEA WebLogic Workshop Workflows

Because Application Explorer runs within BEA WebLogic Workshop, you can easily incorporate Web services into BEA WebLogic Workflows. To enable BEA WebLogic Workshop to use Web services, you export the WSDL to a directory accessible to BEA WebLogic Workshop.

Procedure: How to Export WSDL

To export WSDL to a directory accessible to BEA WebLogic Workshop:

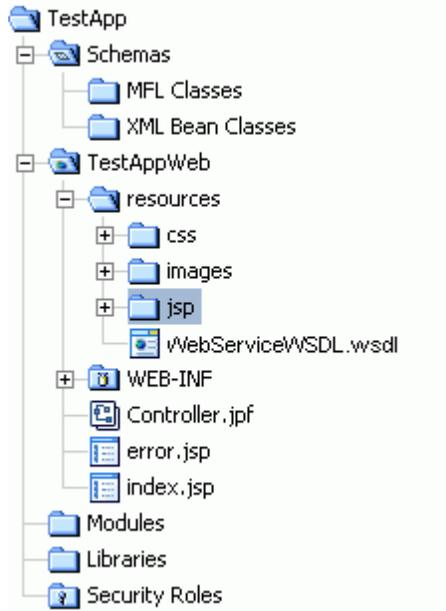
1. After you create a Web service, right-click the Web service name and select *Export WSDL*. The following image is an example of a Web service and its right-click menu.



The Save dialog box appears.

2. Save the WSDL to a directory accessible to BEA WebLogic Workshop, for example, the \resources directory in your BEA WebLogic Workshop Web application directory structure.

The WSDL file appears below the resources folder of your Web application. The following image shows an example of a WSDL file (WebServiceWSDL.wsdl) saved to the resources directory of the TestAppWeb Web application.



Identity Propagation

If you test or execute a Web service using a third party XML editor, for example XMLSPY, the Username and Password values that you specify in the SOAP header must be valid and are used to connect to Oracle. The user name and password values that you provided for Oracle during target creation using Application Explorer are overwritten for this Web service request. The following is a sample SOAP header that is included in the WSDL file for a Web service:

```
<SOAP-ENV:Header>
  <m:ibsinfo xmlns:m="urn:schemas-iwaysoftware-com:iwse">
    <m:service>String</m:service>
    <m:method>String</m:method>
    <m:license>String</m:license>
    <m:disposition>String</m:disposition>
    <m:Username>String</m:Username>
    <m>Password>String</m>Password>
    <m:language>String</m:language>
  </m:ibsinfo>
</SOAP-ENV:Header>
```

Note: You can remove the following tags from the SOAP header, since they are not required:

```
<m:disposition>String</m:disposition>  
<m:language>String</m:language>
```

Adding a Control for a Resource in BEA WebLogic Workshop

Java controls provide a convenient way to incorporate access to other resources. You can add controls in BEA WebLogic Workshop to use Web services created by Application Explorer, or you can add controls that enable you to take advantage of the JCA resources of Application Explorer.

Adding a Web Service Control to a BEA WebLogic Workshop Application

After you create a Web service using Application Explorer and export the WSDL file, you can create a control for the Web service.

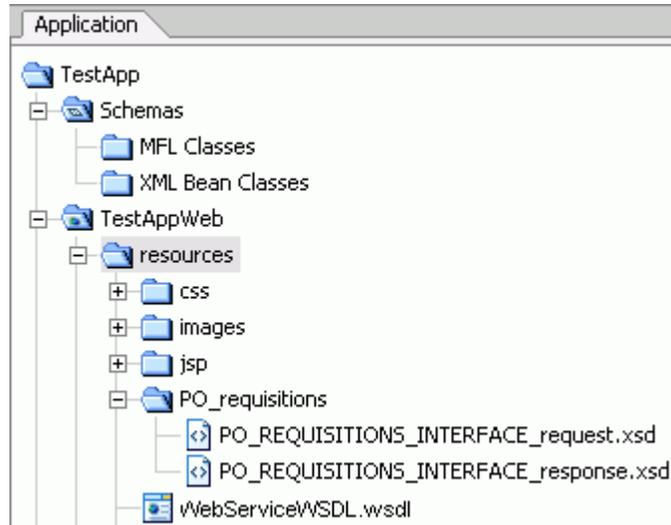
For more information on exporting a WSDL file, see *Export WSDL* on page A-23.

Procedure: How to Add a Web Service Control

To add a Web service control:

1. After exporting the WSDL file from Application Explorer, locate the file in the Application tab of your BEA WebLogic Workshop application.

For example, a WSDL file saved to the \resources directory in your BEA WebLogic Workshop Application directory structure appears, as shown in the following image.



2. Right-click the *WSDL* file and select *Generate Service Control*.

The control for the WSDL appears below the WSDL file in the resources tree, as shown in the following image.



Adding an Extensible CCI Control to a BEA WebLogic Workshop Application

The following section describes the enhanced CCI control, which is extensible and provides JCX with typed inputs and outputs for JCA in BEA WebLogic Workshop.

Overview

The extensible iWay CCI control provides:

- **Method and tag validation.** BEA WebLogic Workshop provides warnings regarding invalid methods and tags.
- **Improved error handling.**

You can define new methods that rely on the generic *service* and *authService* methods. For example, you can define a JCX with a new method without writing casting code or explicit transformations such as the following:

```
public iwaysoftwareAdapterIworacleTableresponse.ORACLEDocument
callOraApp(iwaysoftwareAdapterIworacleTablerequest.ORACLEDocument
aRequest) throws Exception;
```

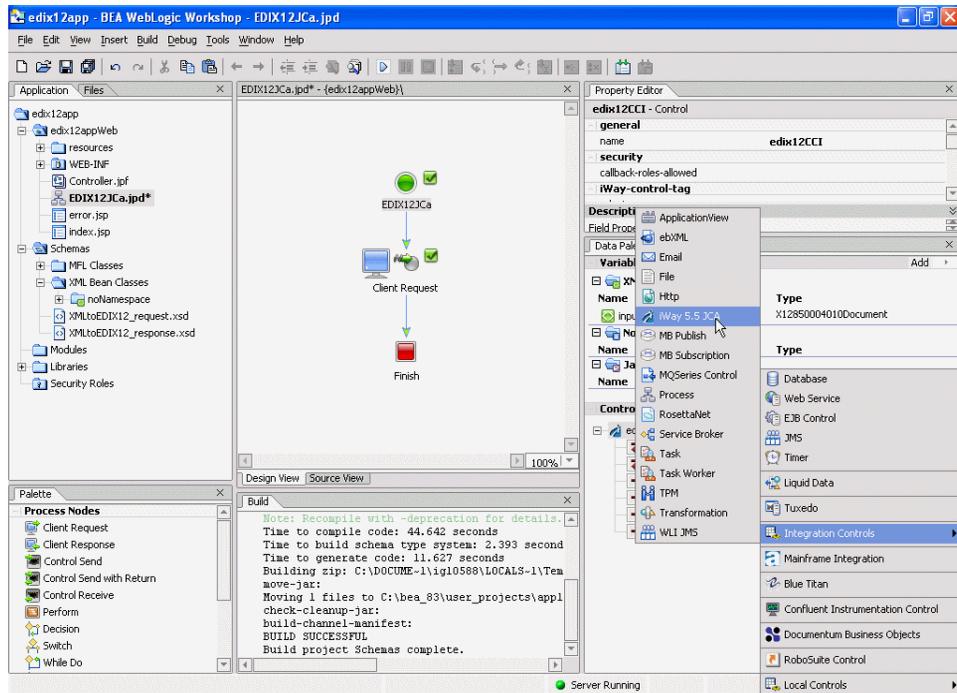
In addition, the extensible CCI control now generates a JCX file to which you can add your own methods. For more information, see *Defining a Control Using the Extensible CCI Control* on page A-27.

You can also use dynamic class casting to specify schema-based input or output XmlObjects to be casted into a pure XmlObject as a service method, which is expected by the CCI control. For more information, see *Using Dynamic Class Casting* on page A-33.

Example: Defining a Control Using the Extensible CCI Control

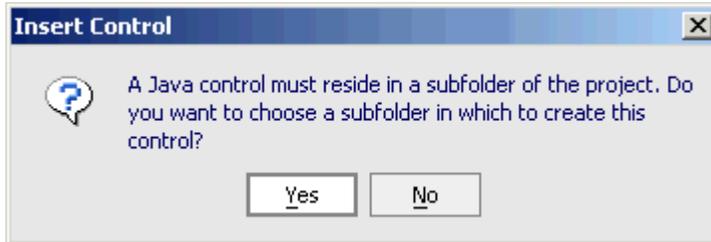
The following sample JCX demonstrates how to define a control for Oracle E-Business Suite using the extensible CCI control in BEA WebLogic Workshop.

1. Start BEA WebLogic Workshop and create a new project. The following image shows an example of a populated WebLogic Workshop screen.



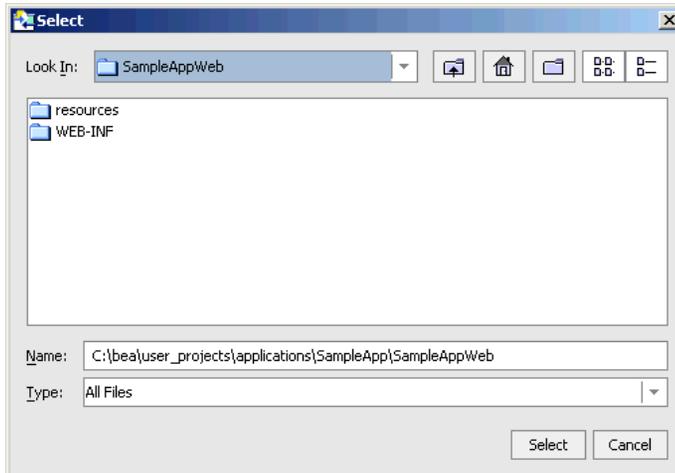
2. Click *Add* from the Controls section in the Data Palette tab, select *Integration Controls*, and click *iWay 5.5 JCA*.

The Insert Control message box opens, as shown in the following image.



3. Click *Yes*.

The Select dialog box opens, as shown in the following image.



4. Choose a subfolder for the CCI control and click *Select*.

The Insert Control - iWay 5.5 JCA dialog box opens. An example of this dialog box, which provides fields for you to define the control, appears in the following image.

Insert Control - iWay 5.5 JCA

STEP 1 Variable name for this control:

STEP 2 I would like to :

Use an iWay 5.5 JCA control already defined by a JCX file

JCX file:

Create a new iWay 5.5 JCA control to use.

New JCX name:

Make this a control factory that can create multiple instances at runtime

STEP 3

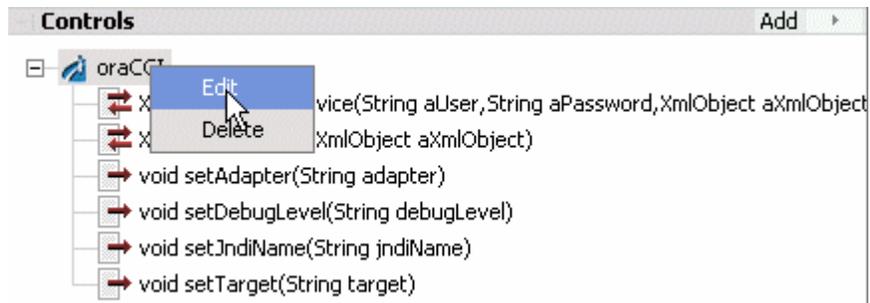
Adapter Name:

Target Name:

Debug Level: ▾

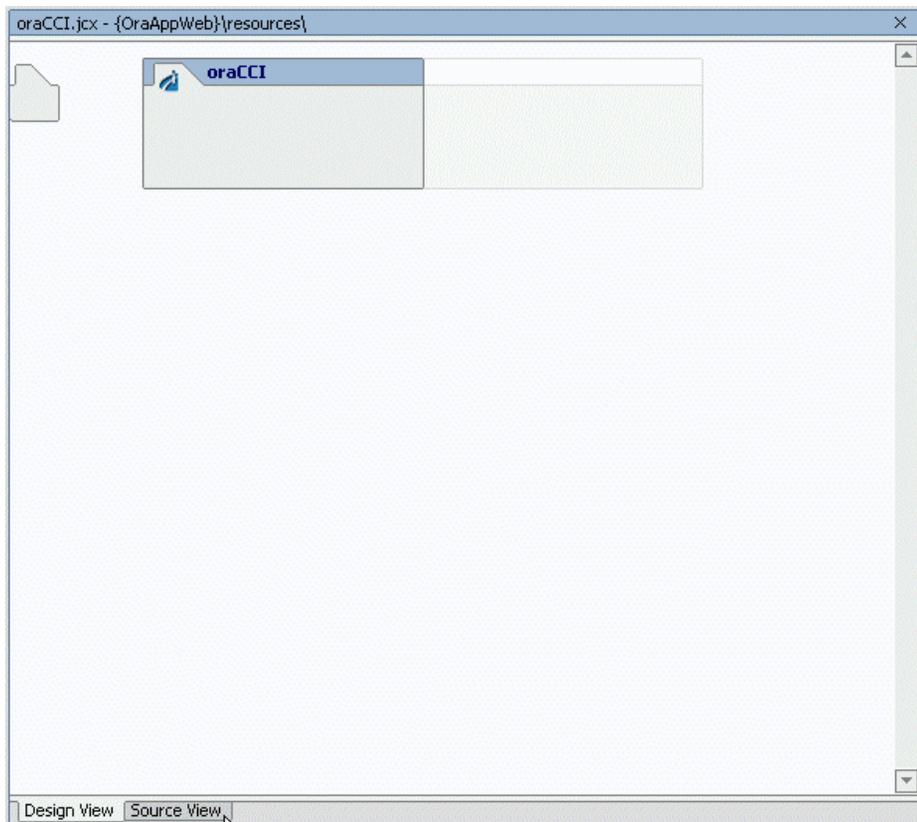
- a. Provide a variable name for the control.
 - b. Click *Create a new iWay 5.5 JCA control to use* and provide a new JCX name.
 - c. Enter the adapter name, target name, and select a debug level from the drop-down list.
5. Click *Create*.

A new JCX file is created. The following image shows the right-click menu of a control.



6. Right-click the control, for example, oraCCI, and select *Edit*.

The Design View for the control opens, as shown in the following image.



7. Click the *Source View* tab.

The Source View for the control opens. An example of a control source view is shown in the following image.

```

oraCCI.jcx* - {OraAppWeb}\resources\
* @jc:iWay-control-tag debugLevel="ERROR" target="oraApp" adapter="Oracle Applicati
*/
public interface oraCCI extends ICCIControl, ControlExtension
{
/**
 * A version number for this JCX. You would increment this to ensure
 * that conversations for instances of earlier versions were invalid.
 */
static final long serialVersionUID = 1L;

// Add you methods here, according to the following examples. You can choose you
// own method names, the adapter uses the number of parameters to determine wheth
// call the service() or the authService() method.

// A call to a basic service only has a single parameter, which
// is a subtype of XmlObject. It returns another XmlObject.
public iwaysoftwareAdapterIworacleTableresponse.ORACLEDocument callOraApp(iwayso

// A call to an authenticated service has two additional parameters
// corresponding to the users credentials.
// public BAPIMATERIALGETDETAILResponseDocument getDetail(String aUser, String a

}

```

Design View | Source View

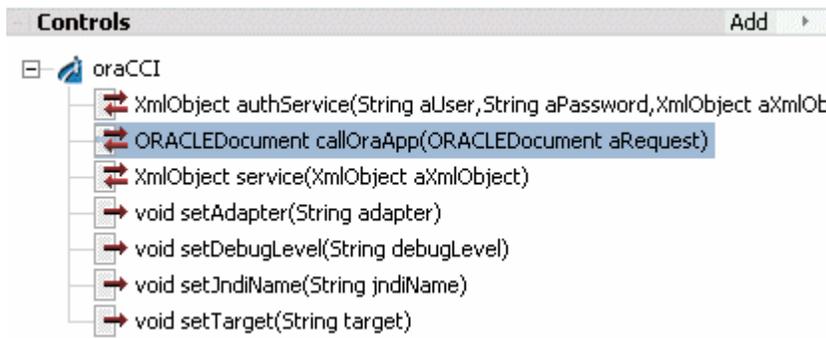
Perform the following steps:

- a. Uncomment the public class definition.
- b. Change the existing response data type to match your response data type according to the Oracle E-Business Suite schema, for example, `iwaysoftwareAdapterIworacleTableresponse.ORACLEDocument`.
- c. Change the existing method name to match your method, for example, `callOraApp`.
- d. Change the existing incoming data type to match your request data type according to the Oracle E-Business Suite schema, for example, `iwaysoftwareAdapterIworacleTablerequest.ORACLEDocument`.

The following syntax provides a sample Oracle E-Business Suite method call, which calls the adapter service:

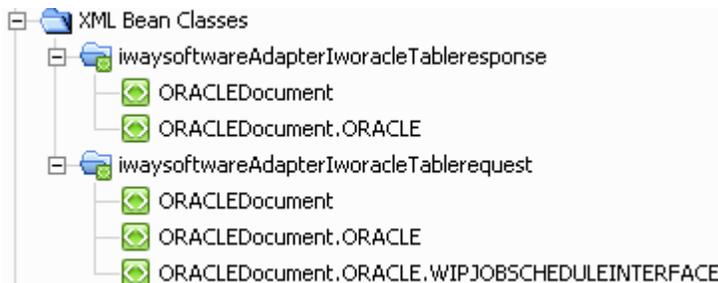
```
public iwaysoftwareAdapterIworacleTablresponse.ORACLEDocument  
callOraApp(iwaysoftwareAdapterIworacleTablerequest.ORACLEDocument  
aRequest) throws Exception;
```

The following control is now available in BEA WebLogic Workshop and can be added to a workflow. The following images shows the newly creaed control in the Controls list.



Note: You can view available data types under the *XML Bean Classes* folder in the *Application* tab, which are added once you import your XML request or response schemas from Application Explorer.

These data types are case sensitive and must be entered exactly as shown.



Using the Extensible CCI Control

The extensible CCI control functions much like a database control since it generates JCX files to which you can add your own methods.

Your own methods can use the correct input and output types rather than the generic XmlObject types that the JCA control uses. Since the control is just a proxy that uses a reflection to call the relevant method, it handles the casting for you. You are no longer required to write custom code that does the cast or transformations that are cast between an XmlObject.

For example, instead of the generic XmlObject:

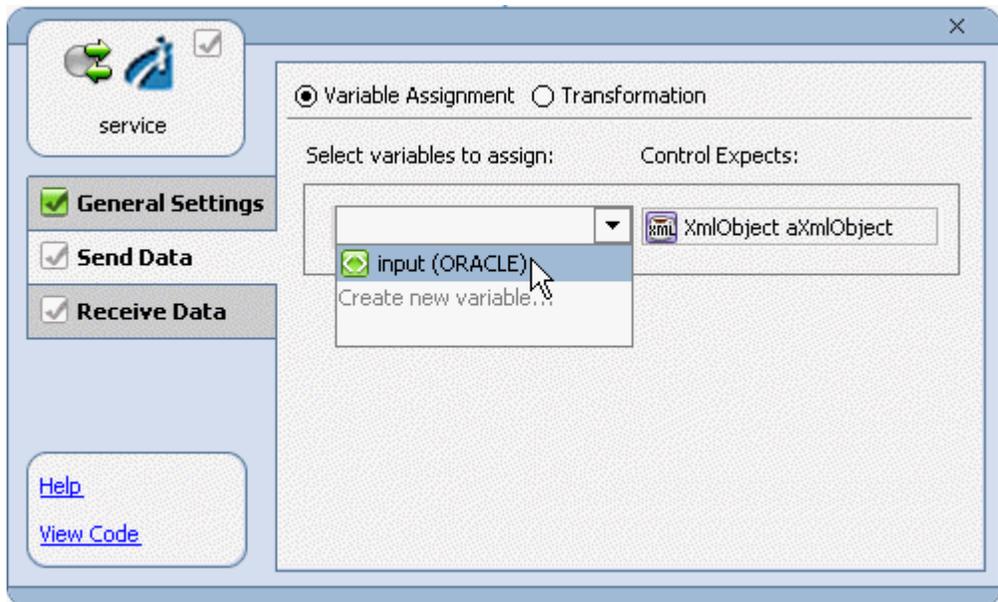
```
XmlObject service(XmlObject input) throws java.lang.Exception;
```

you call:

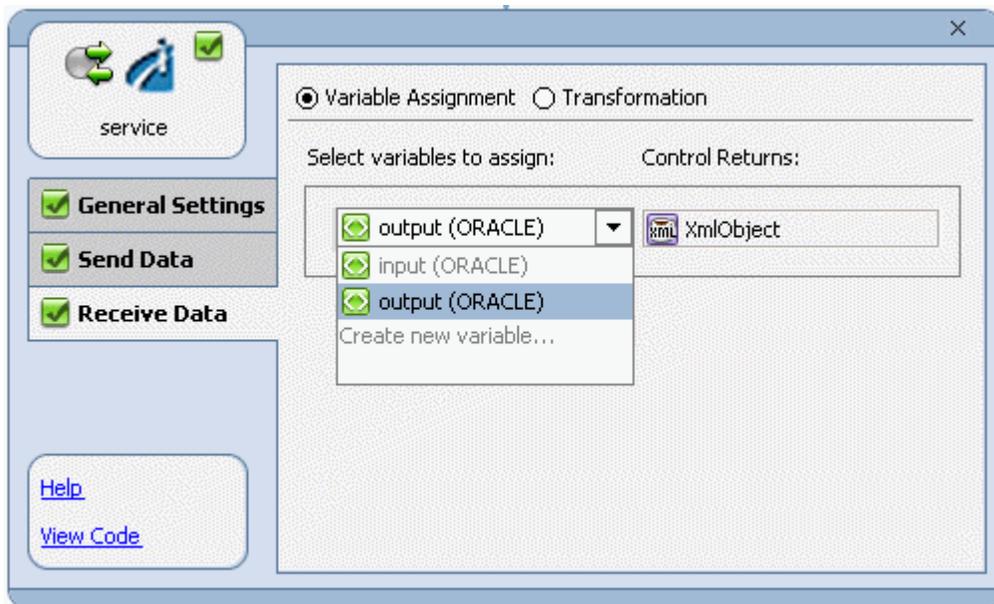
```
public iwaysoftwareAdapterIworacleTablereponse.ORACLEDocument  
callOraApp(iwaysoftwareAdapterIworacleTablerequest.ORACLEDocument  
aRequest) throws Exception;
```

Example: Using Dynamic Class Casting

The following example uses dynamic class casting to specify a schema-based input XmlObject to be casted into a pure XmlObject as a service method, which is expected by the CCI control.



The following example uses dynamic class casting where the CCI control returns a pure XmlObject, which is casted dynamically into a schema-based output XmlObject.



APPENDIX B

Using Application Explorer in BEA WebLogic Workshop for Event Handling

Topics:

- Starting Application Explorer in BEA WebLogic Workshop
- Understanding Event Functionality
- Creating an Event Port
- Modifying an Event Port
- Creating a Channel
- Modifying a Channel
- Deploying Components in a Clustered BEA WebLogic Environment

This section describes how to use Java Swing Application Explorer running in BEA WebLogic Workshop to listen for events in Oracle E-Business Suite.

Starting Application Explorer in BEA WebLogic Workshop

The server must be started on which Application Explorer is running. Before you can use Application Explorer, you must start BEA WebLogic Server.

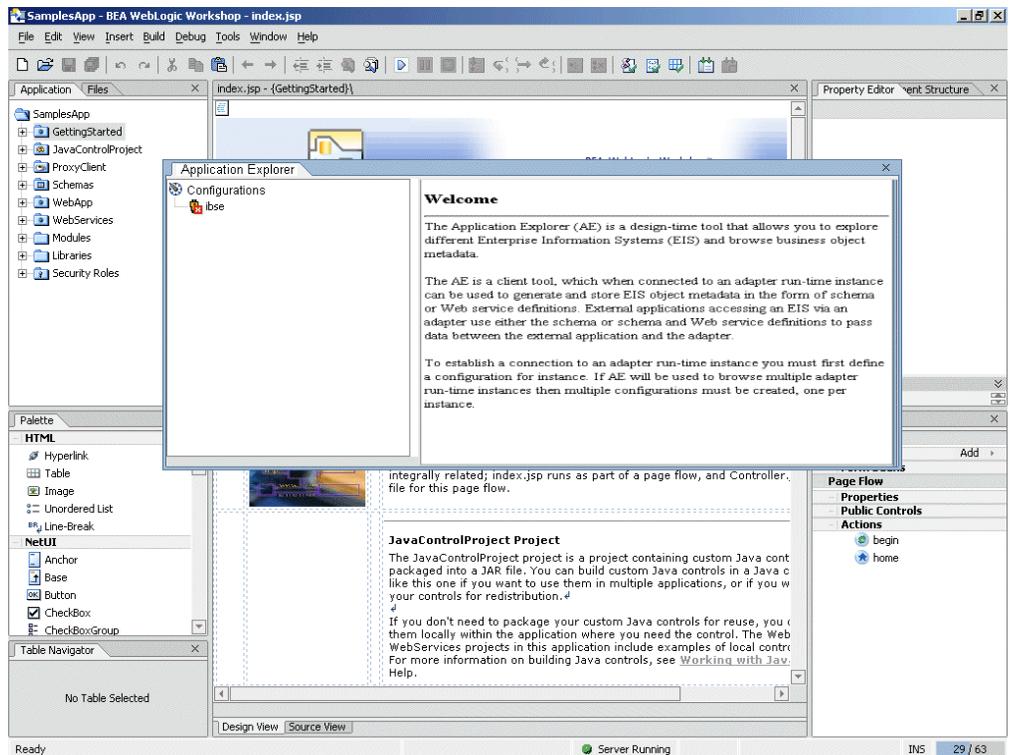
You can run Application Explorer in BEA WebLogic Workshop using an Integration Business Services Engine (IBSE) configuration or J2EE Connector Architecture (JCA) configuration.

Procedure: How to Start Application Explorer in BEA WebLogic Workshop

To start Application Explorer running in BEA WebLogic Workshop:

1. Ensure that the BEA WebLogic Server is running.
2. Start BEA WebLogic Workshop.
3. From the BEA WebLogic Workshop View menu, select *Windows* and then, *Application Explorer*.

Application Explorer opens in BEA WebLogic Workshop as a separate window with a Configurations node on the left and welcome information on the right, as shown in the following image.



You can resize and drag-and-drop the Application Explorer window within BEA WebLogic Workshop. For example, you can drag it to the upper part of BEA WebLogic Workshop.

Understanding Event Functionality

Events are generated as a result of Oracle E-Business Suite activity. You can use events to trigger an action in your application. For example, you can detect WIP discrete job creation and notify another enterprise integration system of the event.

After you create a connection to your application system, you can add events using Application Explorer. To define an Oracle E-Business Suite event to the adapter, you must create a port and a channel.

- Port

A port associates a particular interface table exposed by the adapter with a particular disposition. A disposition defines the protocol and location of the event data. The port defines the end point of the event consumption.

For example, you can use a JMS protocol to route the result of polling an interface table to a JMS queue hosted by BEA WebLogic Server. For more information, see *Creating an Event Port* on page B-3.

- Channel

A channel represents configured connections to particular instances of back-end systems. A channel binds one or more event ports to a particular listener managed by the adapter. For more information, see *Creating a Channel* on page B-23.

There are several techniques that you can employ when listening for Oracle E-Business Suite events, depending upon your requirements. These techniques are described in *Choosing a Listening Technique* in Chapter 3, *Listening for Oracle E-Business Suite Events*.

Creating an Event Port

The following procedures describe how to create an event port using Application Explorer. When you use Application Explorer with an Integration Business Services Engine (iBSE) implementation, the following port dispositions are available:

- File
- iBSE
- MSMQ
- JMS
- HTTP

Creating an Event Port

- SOAP
- MQ Series

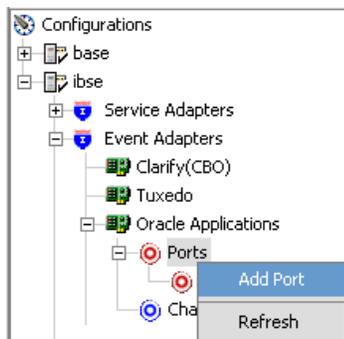
When you use Application Explorer with a JCA implementation, the following port dispositions are available:

- File
- HTTP
- JMS
- MQSeries

Procedure: How to Create an Event Port for File

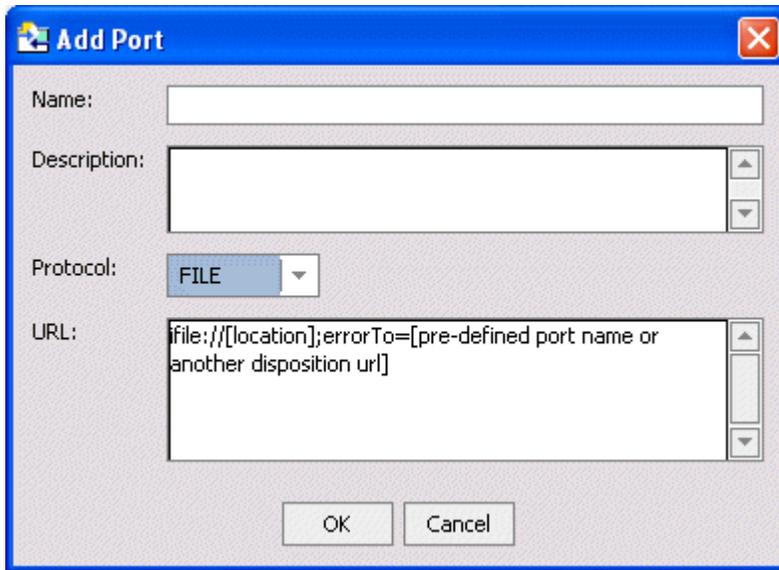
To create an event port for File:

1. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and select *Ports*.



2. Right-click the *Ports* node and select *Add Port*, as shown in the previous image.

The Add Port dialog box opens. This dialog box, shown in the following image, provides four fields to define the new port and two action buttons.



- a. In the Name field, type a name for the event port, for example, OracleFile.
- b. In the Description field, type a brief description.
- c. From the Protocol drop-down list, select *FILE*.
- d. In the URL field, type a URL to a destination file where the event data will be written.

When pointing Application Explorer to an **ibSE** deployment, specify the destination file using the following format:

```
file:///location[;errorTo=errorDest]
```

When pointing Application Explorer to a **JCA** deployment, specify the destination file using the following format:

```
location
```

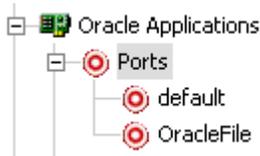
The following table describes the URL parameters.

Parameter	Description
location	The full directory path and file name to which the data is written.

Parameter	Description
errorTo	Location where error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

3. Click OK.

On the left, the event port appears below the Ports node. The following image shows a port named OracleFile under the Ports node.



4. To review the port settings, select the port name.

On the right, a table appears that summarizes the information associated with the event port, as shown in the following image.

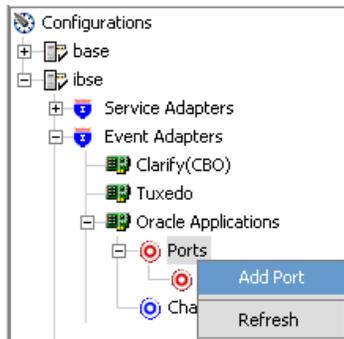
Detail	
Name	Value
Name	OracleFile
Description	sample port
Disposition	file:///[[location]];errorTo=[pre-defined port name or an...
Content	all messages accepted

You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page B-23.

Procedure: How to Create an Event Port for iBSE

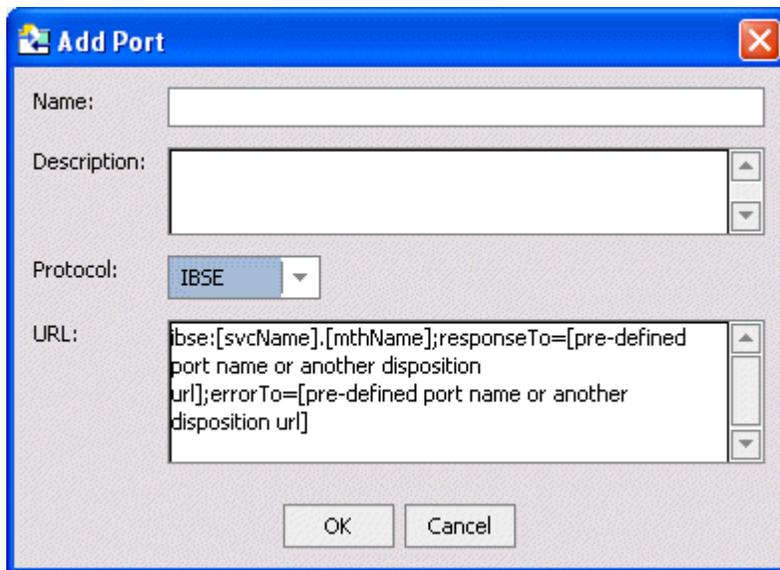
To create an event port for Integration Business Services Engine (iBSE):

1. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and select *Ports*.



2. Right-click the *Ports* node and select *Add Port*.

The Add Port dialog box opens. This dialog box, shown in the following image, provides four fields to define the new port and two action buttons.



- a. In the Name field, type a name for the event port, for example, OracleiBSE.
- b. In the Description field, type a brief description.
- c. From the Protocol drop-down list, select *IBSE*.

- d. In the URL field, type a URL to an iBSE destination using the following format:

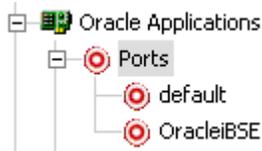
`ibse:[svcName].methName[;responseTo=respDest][;errorTo=errorDest]`

The following table describes the disposition parameters.

Parameter	Description
svcName	Name of the service created with iBSE.
methName	Name of the method created for the Web service.
respDest	Location where responses to the Web service are posted. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.
errorDest	Location where error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

3. Click *OK*.

On the left, the event port appears below the Ports node. The following image shows a port named OracleiBSE under the Ports node.



4. To review the port settings, select the port name.

On the right, a table appears that summarizes the information associated with the event port, as shown in the following image.

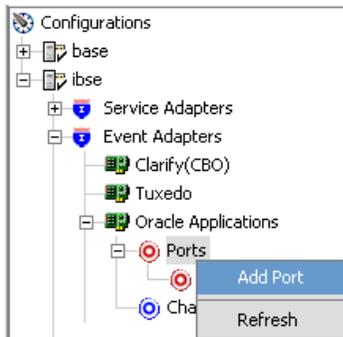
Detail	
Name	Value
Name	OracleiBSE
Description	example iBSE Port
Disposition	ibse:[svcName].[methName];responseTo=[pre-defined...
Content	all messages accepted

You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page B-23.

Procedure: How to Create an Event Port for MSMQ

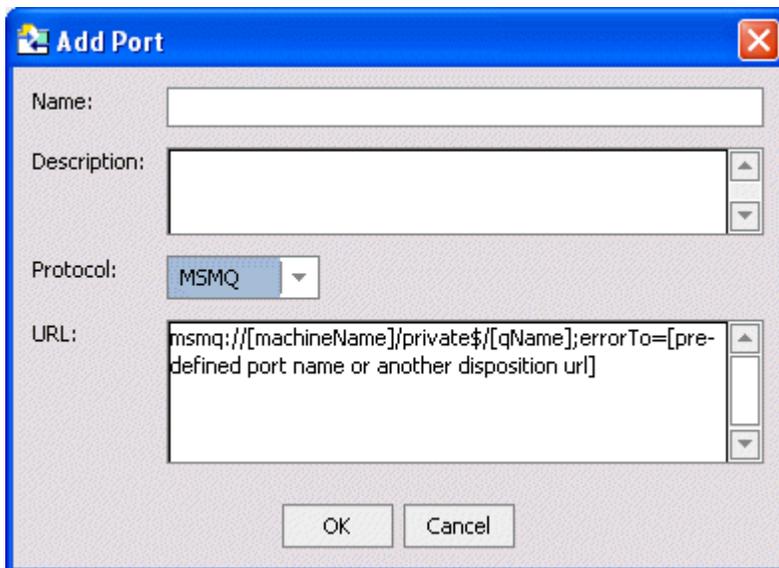
To create an event port for a Microsoft Message Queuing (MSMQ) queue:

1. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and select *Ports*.



2. Right-click the *Ports* node and select *Add Port*.

The Add Port dialog box opens. This dialog box, shown in the following image, provides four fields to define the new port and two action buttons.



- a. In the Name field, type a name for the connection, for example, OracleMSMQ.
- b. In the Description field, type a description for the target name you just created.
- c. From the Protocol drop-down list, select *MSMQ*.

- d. In the URL field, type a URL to an MSMQ destination in the following format:

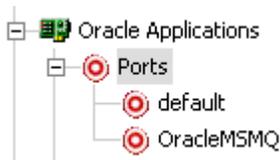
`msmq: /host/queueType/queueName[;errorTo=errorDest]`

The following table defines the disposition parameters.

Parameter	Description
host	Name of the machine on which the Microsoft Queuing system runs.
queueType	The type of queue. For private queues, enter <i>Private\$</i> . Private queues are queues that are not published in Active Directory. They appear only on the local computer that contains them. Private queues are accessible only by Message Queuing applications that recognize the full path name or format name of the queue.
queueName	Name of the queue where messages are placed.
errorDest	Location where error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

3. Click OK.

On the left, the event port appears below the Ports node. The following image shows a port named OracleMSMQ under the Ports node.



4. To review the port settings, select the port name.

On the right, a table appears that summarizes the information associated with the event port, as shown in the following image.

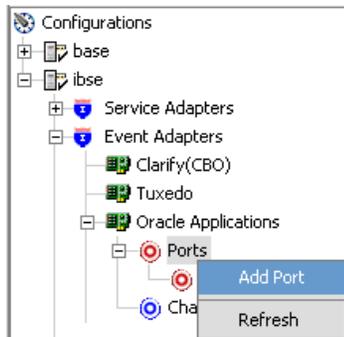
Detail	
Name	Value
Name	OracleMSMQ
Description	
Disposition	msmq://[machineName]/private\$/[qName];errorTo=[p...
Content	all messages accepted

You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page B-23.

Procedure: How to Create an Event Port for JMS

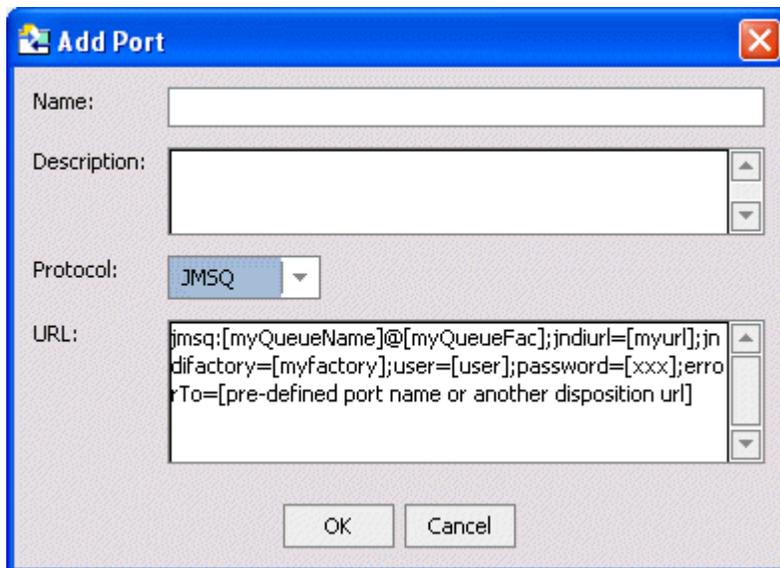
To create an event port for a JMS queue:

1. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and select *Ports*.



2. Right-click the *Ports* node and select *Add Port*.

The Add Port dialog box opens. This dialog box, shown in the following image, provides four fields to define the new port and two action buttons.



- a. In the Name field, type a name for the event port, for example, OracleJMSQ.

- b. In the Description field, type a brief description.
- c. From the Protocol drop-down list, select *JMSQ*.
- d. In the URL field, type a URL to a JMSQ destination file.

When pointing Application Explorer to an **ibSE** deployment, specify the destination file using the following format:

```
jmsq:queue@conn_factory;jndiurl=jndi_url;jndifactory=jndi_factory;
user=userID;password=pass[;errorTo=errorDest]
```

When pointing Application Explorer to a **JCA** deployment, specify the destination file using the following format:

```
jms:queue@conn_factory;jndiurl=jndi_url;jndifactory=jndi_factory
```

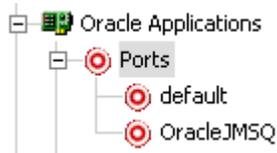
The following table describes the URL parameters.

Parameter	Description
queue	Name of a queue to which events are emitted.
conn_factory	The connection factory, a resource that contains information about the JMS Server. The BEA WebLogic connection factory is: <code>javax.jms.QueueConnectionFactory</code>
jndi_url	The URL of the application server. For BEA WebLogic Server, the URL is <code>t3://host:port</code> where: <code>host</code> Is the machine name where BEA WebLogic Server resides. <code>port</code> Is the port on which BEA WebLogic Server is listening. The default port, if not changed at installation, is 7001.
jndi_factory	Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider. For BEA WebLogic Server, the BEA WebLogic factory is <code>weblogic.jndi.WLInitialContextFactory</code> .
userID	User ID associated with this queue.
pass	Password associated with this user ID.

Parameter	Description
errorDest	Location where error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

3. Click OK.

On the left, the event port appears below the Ports node. The following image shows a port named OracleJMSQ under the Ports node.



4. To review the port settings, select the port name.

On the right, a table appears that summarizes the information associated with the event port, as shown in the following image.

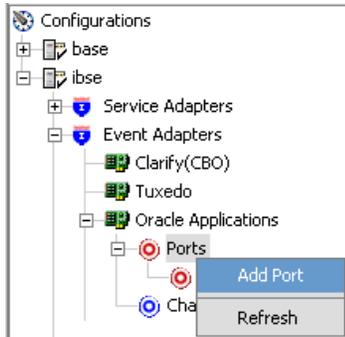
Detail	
Name	Value
Name	OracleJMSQ
Description	example JMSQ port
Disposition	jmsq:[myQueueName]@[myQueueFac];jndiurl=[myurl...
Content	all messages accepted

You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page B-23.

Procedure: How to Create an Event Port for SOAP

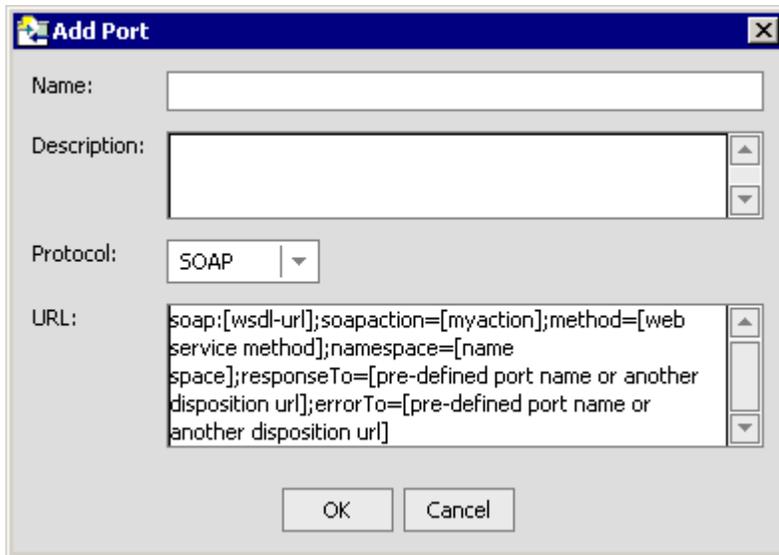
To create a port for an SOAP disposition:

1. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and select *Ports*.



2. Right-click the *Ports* node and select *Add Port*.

The Add Port dialog box opens. This dialog box, shown in the following image, provides four fields to define the new port and two action buttons.



- a. In the Name field, type a name for the event port, for example, OracleSOAP.
- b. In the Description field, type a brief description.
- c. From the Protocol drop-down list, select SOAP.

- d. In the URL field, type a URL for a SOAP destination, using the following format:

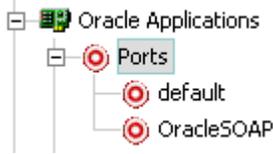
```
soap: [wsdl-url];soapaction=[myaction];method=[web service
method];namespace=[namespace];responseTo=[pre-defined port name or
another disposition URL];errorTo=[pre-defined port name or another
disposition url]
```

The following table defines the parameters for the URL.

Parameter	Description
wsdl-url	<p>The URL to the WSDL file that is required to create the SOAP message. For example:</p> <pre>http://localhost:7001/ibse/IBSEServlet/test/ webservice.ibs?wsdl</pre> <p>where:</p> <pre>webservice</pre> <p>Is the name of the Web service you created using Application Explorer.</p> <p>To find this value, navigate to the Integration Business Services tab, expand the <i>Services</i> node, select the service you created, and click the <i>Service Description</i> link on the right. The WSDL URL appears in the Address field of the window that opens.</p> <p>You can also open the WSDL file in a third party XML editor (for example, XMLSPY) and view the SOAP request settings to find this value.</p>
soapaction	The method that will be called by the SOAP disposition. This value can be found in the WSDL file.
method	The Web service method you are using. This value is found in the WSDL file.
namespace	The XML namespace you are using. This value is found in the WSDL file.
responseTo	The location to which responses are posted. A predefined port name or another full URL. Optional. The URL must be complete, including the protocol.
errorTo	<p>The location to which error logs are sent. Optional.</p> <p>A predefined port name or another disposition URL. The URL must be complete, including the protocol.</p>

3. Click *OK*.

On the left, the event port appears below the Ports node. The following image shows a port named OracleSOAP under the Ports node.



4. To review the port settings, select the port name.

On the right, a table appears that summarizes the information associated with the event port, as shown in the following image.



Name	Value
Name	OracleSOAP
Description	example SOAP port
Disposition	soap:[wsdl-url];soapaction=[myaction];method=[web...
Content	all messages accepted

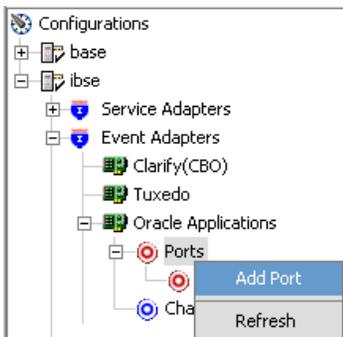
You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page B-23.

Procedure: How to Create an Event Port for HTTP

The HTTP disposition uses an HTTP URL to specify an HTTP end point to which the event document is posted.

To create an event port for HTTP disposition:

1. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and select *Ports*.



2. Right-click the *Ports* node and select *Add Port*.

The Add Port dialog box opens. This dialog box, shown in the following image, provides four fields to define the new port and two action buttons.

- a. In the Name field, type a name for the event port, for example, OracleHTTP.
- b. In the Description field, type a brief description.
- c. From the Protocol drop-down list, select *HTTP*.
- d. In the URL field, type a URL for an HTTP destination file.

When pointing Application Explorer to an **ibSE** deployment, specify the destination file using the following format:

```
ihttp://url;responseTo=respDest
```

When pointing Application Explorer to a **JCA** deployment, specify the destination file using the following format:

```
http://host:port/uri
```

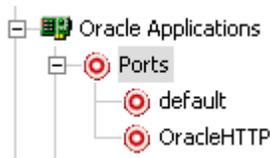
The following table describes the URL parameters.

Parameter	Description
url	The URL target for the post operation.

Parameter	Description
respDest	Location where responses are posted. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.
host	Name of the host on which the Web server resides.
port	Port number on which the Web server is listening.

3. Click OK.

On the left, the event port appears below the Ports node. The following image shows a port named OracleHTTP under the Ports node.



4. To review the port settings, select the port name.

On the right, a table appears that summarizes the information associated with the event port, as shown in the following image.

Detail	
Name	Value
Name	OracleHTTP
Description	example HTTP port
Disposition	http://[myurl];responseTo=[pre-defined port name or ...
Content	all messages accepted

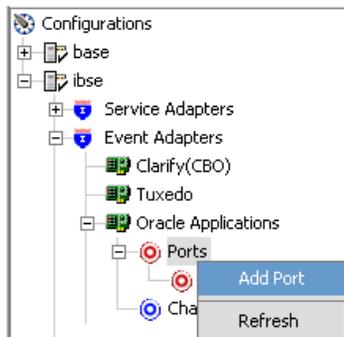
You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page B-23.

Procedure: How to Create an Event Port for MQ Series

The MQSeries disposition allows an event to be enqueued to an MQSeries queue. You can specify both queue manager and queue name.

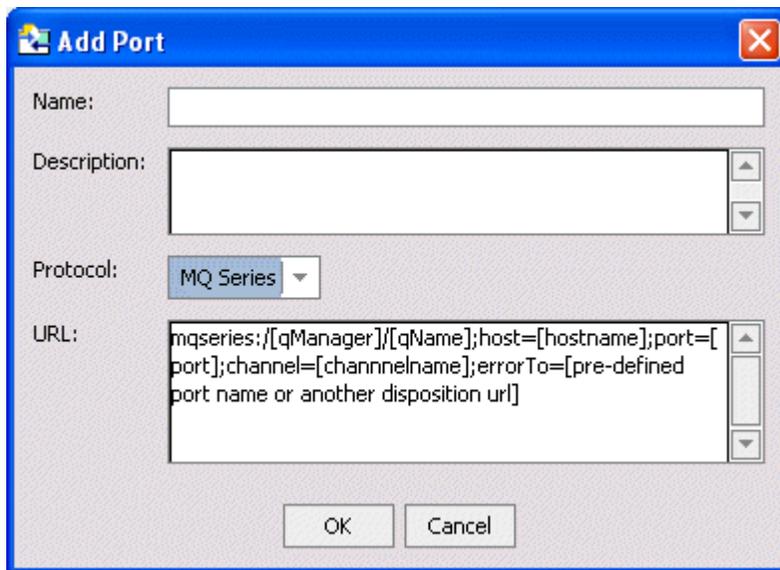
To create a port for an MQSeries queue:

1. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and select *Ports*.



2. Right-click the *Ports* node and select *Add Port*.

The Add Port dialog box opens. This dialog box, shown in the following image, provides four fields to define the new port and two action buttons.



- a. In the Name field, type a name for the event port, for example, OracleMQSeries.

- b. In the Description field, type a brief description.
- c. From the Protocol drop-down list, select *MQSeries*.
- d. In the URL field, type a URL to an MQSeries destination file.

When pointing Application Explorer to an **ibSE** deployment, specify the destination file using the following format:

```
mqseries:/qManager/qName;host=hostName;port=portNum;
channel=chanName[;errorTo=errorDest]
```

When pointing Application Explorer to a **JCA** deployment, use the following format:

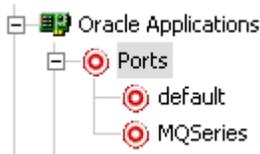
```
mq:qmanager@respqueue;host=;port=;channel=
```

The following table describes the URL parameters.

Parameter	Description
qManager	Name of queue manager to which the server must connect.
qName	Name of the queue where messages are placed.
hostName	Name of the host on which MQSeries resides (MQ client only).
portNum	Port number for connecting to an MQ Server queue manager (MQ client only).
chanName	Case-sensitive name of the channel that connects with the remote MQ Server queue manager (MQ client only). The default MQSeries channel name is SYSTEM.DEF.SVRCONN.
errorDest	Location where error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

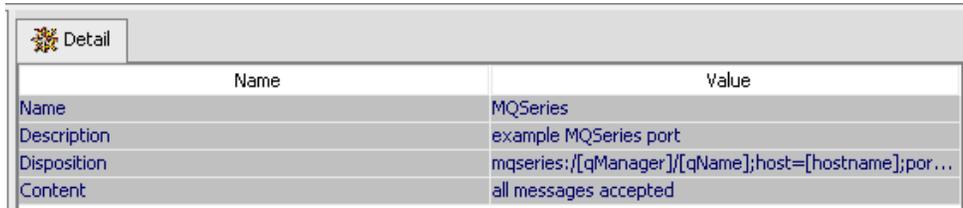
- 3. Click *OK*.

On the left, the event port appears below the Ports node. The following image shows a port named OracleMQSeries under the Ports node.



- 4. To review the port settings, select the port name.

On the right, a table appears that summarizes the information associated with the event port, as shown in the following image.



Detail	
Name	Value
Name	MQSeries
Description	example MQSeries port
Disposition	mqseries;/[qManager]/[qName];host=[hostname];por...
Content	all messages accepted

You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page B-23.

Modifying an Event Port

The following procedures describe how to edit and delete an event port using Application Explorer. If you want to review the port settings, select the port name. The right pane displays summary information associated with that event port.

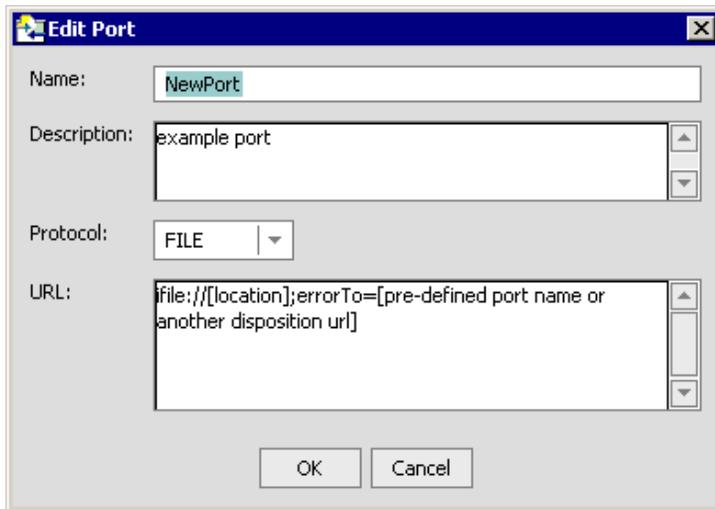
Procedure: How to Edit an Event Port

To edit an event port:

5. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and select *Ports*.
6. Right-click the port you want to edit, and select *Edit*.

Modifying an Event Port

The Edit Port dialog box opens, as shown in the following image, with fields to modify the port parameters and two action buttons.



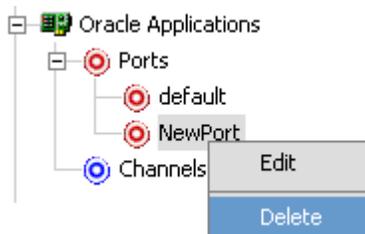
You can not edit the port name.

7. Make the required changes and click *OK*.

Procedure: How to Delete an Event Port

To delete an event port:

1. To view the available ports, click the *Ports* node in the left pane.



2. Right-click the port you want to remove, and select *Delete*, as shown in the previous image.

The event port node disappears from the ports list in the left pane.

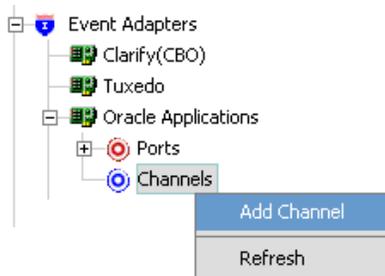
Creating a Channel

The following procedure describes how to create a channel for an Oracle E-Business Suite event. All defined event ports must be associated with a channel.

Procedure: How to Create a Channel

To create a channel for File:

1. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and select *Channels*.



2. Right-click the *Channels* node and select *Add Channel*.

Creating a Channel

The Add Channel dialog box opens, as shown in the following image, with available fields to define the new channel and two action buttons.

The image shows a Windows-style dialog box titled "Add Channel". It has a standard title bar with a close button. The dialog is divided into several sections. At the top, there is a "Name:" label followed by a single-line text input field. Below that is a "Description:" label followed by a larger, multi-line text input field. The next section is "Protocol:" with a dropdown menu currently showing "TABLE Listener". The bottom half of the dialog is split into two columns. The left column is titled "Available Port(s)" and contains a list box with four items: "NewPort", "OracleFile", "OracleIBSE", and "OracleMQSeries". The right column is titled "Selected Port(s)" and contains an empty list box. Between these two list boxes are four buttons: a double-right arrow (>>), a single-right arrow (>), a single-left arrow (<), and a double-left arrow (<<). At the very bottom of the dialog are two buttons: "Next" and "Cancel".

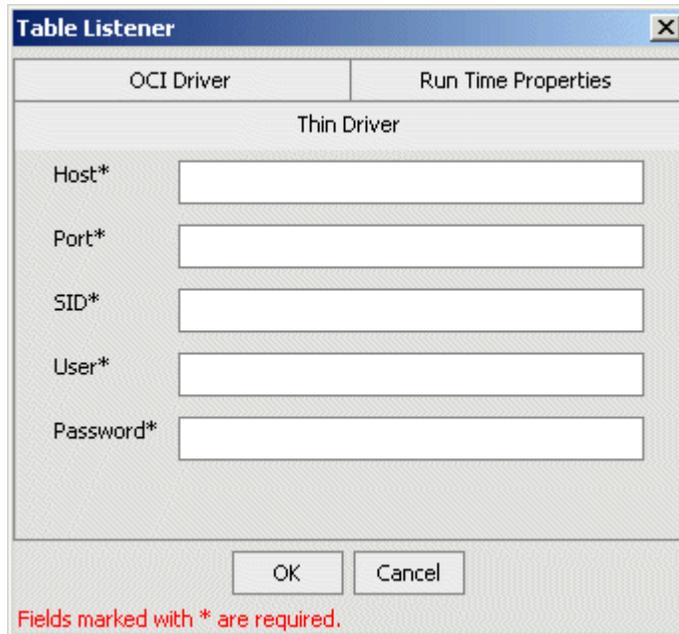
- a. In the Name field, type a name for the channel.
- b. In the Description field, type a brief description.
- c. From the Protocol drop-down list, accept the default, *TABLE listener*.
- d. Associate one or more available ports with this channel:

To move one or more individual port(s) from the Available Port(s) box to the Selected Port(s) box, select the port(s) and click the single-arrow (>) button. Use the Ctrl key to select more than one port.

To move all listed ports from one box to another, click the double-arrow button (>> or <<).

3. Click *Next*.

The Table Listener dialog box opens with the fields appearing for the Thin Driver, and two tabs for the OCI Driver and Run Time Properties, as shown in the following image.



If your Oracle system uses the Oracle JDBC Thin Driver, specify the following information on the Thin Driver tab:

Field	Description
Host	Name of the server on which the Oracle E-Business Suite database instance resides.
Port	Port number on which the database is listening.
SID	A unique name of the database service, chosen by the database administrator or the person who installed Oracle E-Business Suite.
User	The Oracle database user ID to access the Oracle database underlying the Oracle E-Business Suite system. The user ID must have database access to the interface tables being accessed.
Password	Password associated with the specified user ID.

If your Oracle system uses the Oracle JDBC OCI driver, click the OCI Driver tab and specify the following information.

Creating a Channel

Field	Description
Connection TNS Name	<p>Logical name of the Oracle E-Business Suite database instance. (You can obtain the name from the database administrator.)</p> <p>The TNS name that is found in the tnsnames.ora file that contains an entry for that database and that <i>resides on the server running the adapter</i>.</p> <p>If a TNS name does not exist, you must create it. For information about creating a TNS name, see your Oracle administrator.</p>
Release	<p>The version of the Oracle database that supports the Oracle E-Business Suite instance. Select the version from the list of releases supported by the adapter.</p>
User	<p>Oracle database user ID to access the Oracle database underlying the Oracle E-Business Suite system. The user ID must have database access to the interface tables being accessed.</p>
Password	<p>Password associated with the specified user ID.</p>

To enter information in the Run Time Properties tab, use the following information.

Field	Description
Polling Interval (ms)	The interval, in milliseconds, at which to check for new input.
SQL Query	<p>SQL SELECT statement that the listener issues to poll the table.</p> <p>If the statement is omitted, it defaults to</p> <pre>SELECT * FROM table</pre> <p>where:</p> <pre>table</pre> <p>Is the value of the Node Name property.</p> <p>If the SQL statement includes a date column or long text column, you must provide a value for the Post Query property. The value you provide must not contain a date column or a long text column. This applies whether you provide an SQL statement here or rely upon the default.</p> <p>For example, the following SELECT statement retrieves all unprocessed records from the DISCRETE_JOBS table:</p> <pre>SELECT * FROM WIP_DISCRETE_JOBS DJ WHERE DJ.WIP_ENTITY_ID > (SELECT WIP_ENTITY_ID FROM WIP.TEMP_NEW_WORK_ORDER_ENTITY_ID)</pre>

Field	Description
Post Query*	<p>One or more SQL statements that are executed after each new record is read from the table. Separate multiple statements with a semicolon (;).</p> <p>Case sensitive: the case used to specify the column names must match the case used in the SELECT statement that polled the table. If the SQL Query property was omitted so that a default SELECT statement polled the table, the case used to specify the column names must match the case used to define the columns in the database schema.</p> <p>*This property and Issue Post Query Delete are mutually exclusive: you must select one or the other, but not both.</p> <p>If you do not specify a value for Post Query, you must select the Issue Post Query Delete property, which specifies that the adapter deletes each record after it is read. (If you want to retain the table data after it is read, specify a value for Post Query and do not select Issue Post Query Delete.)</p> <p>There are two field operators, ? and ^, that you can use in a post-query SQL statement. For more information, see <i>Post Query Parameter Operators</i> on page B-29.</p>
Node Name	Name of the database table to query with the default SELECT statement if the SQL Query property is omitted. Ignored if SQL Query is specified.
Issue Post Query Delete*	<p>If you select this option, the adapter deletes each record after it is read.</p> <p>*This property and Post Query are mutually exclusive: you must select one or the other, but not both.</p>

4. Click **OK**.

In the left pane, the channel appears below the Channels node.



An X over the icon indicates that the channel is currently disconnected. You must start the channel to activate your event configuration.

5. Select the channel to see detailed information about the channel on the right, as shown in the following image.

	Detail	Thin Driver	OCI Driver	Run Time Properties
Name	NewChannel			
Description				
Type	TABLE Listener			
ports				
Name 0	NewPort			
Name 1	OracleFile			

6. To activate your event configuration, right-click the channel node and select *Start*.

Reference: Post Query Parameter Operators

When you configure an RDBMS Table Listener for Oracle E-Business Suite, you can use two special field operators, `?` and `^`, with the Post Query parameter. Both of these operators dynamically substitute database values in the SQL post-query statement at run time.

- `?fieldname` is evaluated at run time as `field = value`

The `?` operator is useful in UPDATE statements:

```
UPDATE table WHERE ?field
```

For example, the following statement

```
UPDATE Stock_Prices_Temp WHERE ?RIC
```

might be evaluated at run time as:

```
UPDATE Stock_Prices_Temp WHERE RIC = 'PG'
```

- `^fieldname` is evaluated at run time as `value`

The `^` operator is useful in INSERT statements:

```
INSERT INTO table VALUES (^field1, ^field2, ^field3, ... )
```

For example, the following statement

```
INSERT INTO Stock_Prices_Temp VALUES (^RIC, ^Price, ^Updated)
```

Modifying a Channel

might be evaluated at run time as:

```
INSERT INTO Stock_Prices_Temp VALUES ('PG', 88.62, '2003-03-18  
16:24:00.0')
```

Modifying a Channel

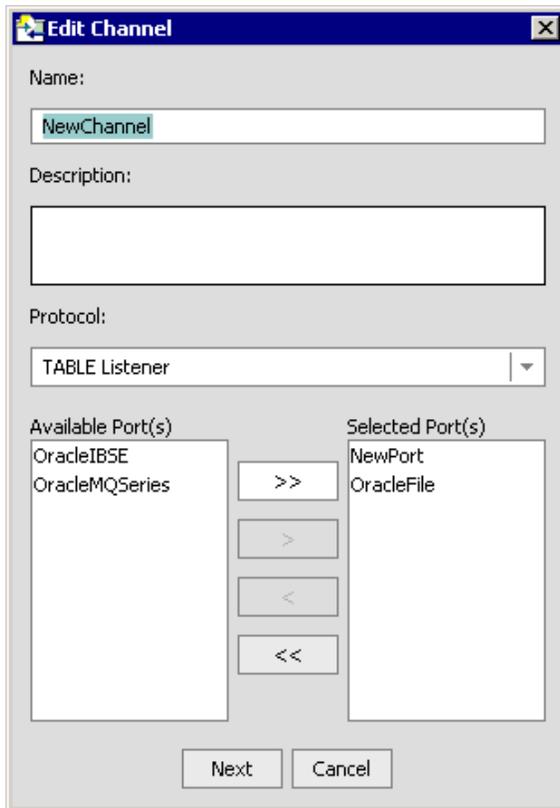
The following procedures describe how to edit and delete a channel through Application Explorer. To review the channel settings, select the channel name. The right pane displays summary information associated with that channel.

Procedure: How to Edit a Channel

To edit a channel:

1. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and select *Channels*.
2. Right-click the channel you want to edit and select *Edit*.

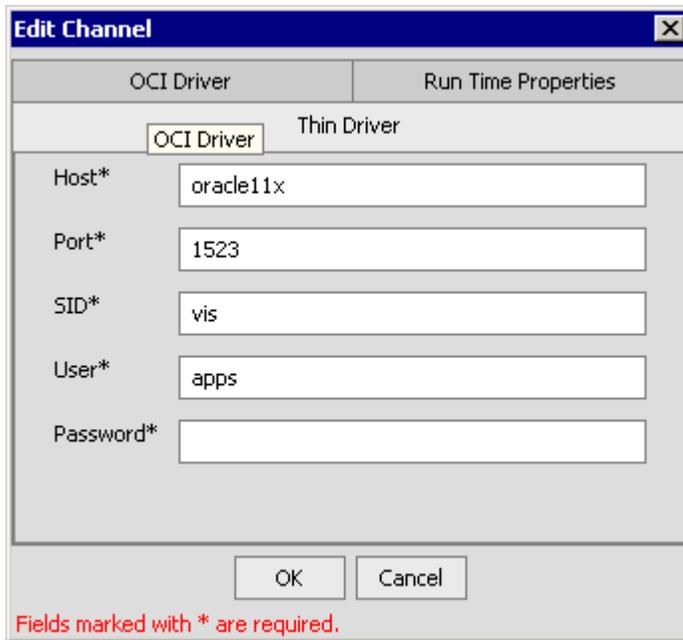
The Edit Channel dialog box opens that allows you to change the associated ports, as shown in the following image.



3. Make the required changes to the channel configuration.
4. Click *Next*.

Modifying a Channel

Another Edit Channel dialog box opens that allows you to change the driver or run time properties information, as shown in the following image.

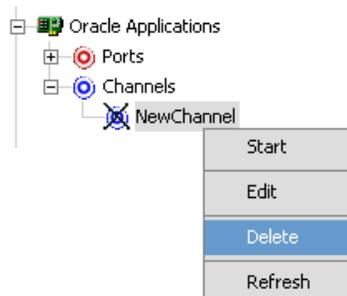


5. Make the required changes and click *OK*.

Procedure: How to Delete a Channel

To delete an existing channel:

1. In the left pane, expand the *Event Adapters* node, the *Oracle Applications* node, and the *Channels* node.



2. Right-click the channel you want to delete and Select *Delete*, as shown in the previous image.

The channel disappears from the Channels list.

Deploying Components in a Clustered BEA WebLogic Environment

Events can be configured in a clustered BEA WebLogic environment. You can deploy iBSE or JCA to this environment. This topic uses iBSE as an example, but you can follow the same procedures when deploying JCA. The only difference is that you need to deploy the JCA connector .RAR file to the clustered environment.

A cluster consists of multiple server instances running simultaneously, yet appears to clients to be a single server instance. The server instances that contain a cluster can be run on one machine, but are usually run on multiple machines.

Clustering provides the following benefits:

- Load balancing
- High availability

Service requests are processed through the HTTP router and routed to an available managed server.

Events are server-specific and are not processed through the HTTP router. You must configure each server separately.

Procedure: How to Deploy Components in a Clustered Environment

To deploy components in a clustered environment:

1. Using the BEA Configuration Wizard:
 - a. Configure an administrative server to manage the managed servers.
 - b. Add and configure as many managed servers as required.
 - c. Add and configure an HTTP router. This does not have to be a part of BEA WebLogic and can be an outside component.
 - d. If you configure the HTTP router within BEA WebLogic, start it by entering the following command:

```
StartManagedWebLogic HTTPROUTER http://localhost:7001
```

where:

```
HTTPROUTER
```

Is the name of the server on which the HTTP router is running.

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

Is the location of the admin console.

- e. Add the managed servers to your cluster/clusters.

For more information on configuring BEA WebLogic Integration for deployment in a clustered environment, see *Deploying WebLogic Integration Solutions*.

2. Start BEA WebLogic Server and open BEA WebLogic Server Console.
3. Deploy iBSE to the cluster by selecting *Web Application Modules* from the Domain Configurations section, and clicking *Deploy a new Web Application Module*.

A page appears for you to specify where the Web application is located.

4. To deploy iBSE, select the option button next to the *ibse* directory and then click *Target Module*.

Deploy a Web Application Module

Select the archive for this Web application module

Select the file path that represents your archive or exploded archive directory.

Note: Only valid file paths are shown below. If you do not find what you are looking for, [click your file\(s\)](#) and/or confirm your Web application module contains valid descriptors.

Location: [localhost](#) \ [C:](#) \ [iWay55](#) \ [bea](#)

<input checked="" type="radio"/>	ibse
<input type="radio"/>	iwaie
<input type="radio"/>	iwjcaivp

5. To deploy Servlet Application Explorer, select the option button next to the *iwaie* directory then click *Target Module*.

If you are using Servlet Application Explorer, deploy it only on the admin server or one of the managed servers.

Deploy a Web Application Module

Select the archive for this Web application module

Select the file path that represents your archive or exploded archive directory.

Note: Only valid file paths are shown below. If you do not find what you are looking for, you should [upload your file\(s\)](#) and/or confirm your Web application module contains valid descriptors.

Location: [localhost \ C: \ Program Files \ iWay55 \ bea](#)

<input type="radio"/>	ibse
<input checked="" type="radio"/>	iwa
<input type="radio"/>	iwjcaivp

Target Module

The following window opens.

Select targets for this Web application module

Select the servers and/or clusters on which you want to deploy your new Web Application module

Independent Servers

AdminServer

HTTPROUTER

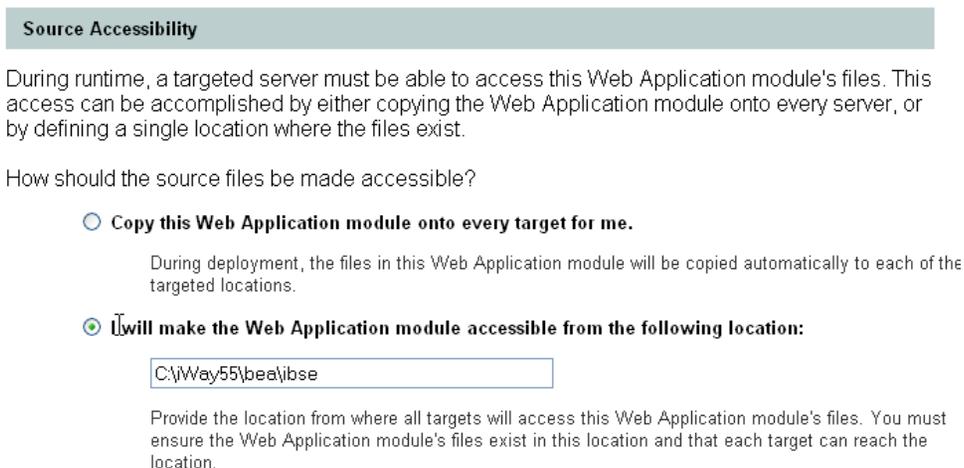
Clusters

MYCluster

- All servers in the cluster
- Part of the cluster
 - MS1
 - MS2

6. Select the servers and/or clusters on which you want to deploy the application and click *Continue*.

The following window opens.



7. Select the *I will make the Web Application module accessible from the following location* option button and provide the location from which all targets will access iBSE.

We recommend that you use a single instance of iBSE, rather than copying iBSE onto every target.

Note: iBSE must use a database repository (SQL or Oracle). Do not use a file repository. You can select this in the Repository Type drop-down list in the iBSE monitoring page. After configuring a database repository, you must restart all of the managed servers.

<http://hostname:port/ibse/IBSEConfig/>

where:

hostname

Is where your application server is running. Use the IP address or machine name in the URL; do not use localhost.

port

Is the port specific to each server, since you deploy iBSE to an entire cluster. For example, 8001, 8002, or any other port that is specified for each managed node.

8. Click *Deploy*.

Procedure: How to Configure Ports and Channels in a Clustered Environment

You can use Swing Application Explorer deployed in BEA WebLogic WorkShop or Servlet Application Explorer to configure ports and channels in a clustered environment.

Note: Before using Servlet Application Explorer in a clustered environment, you must edit the web.xml file and specify the correct URL to your iBSE deployment. The default location on Windows is:

`C:\Program Files\iWay55\bea\iwaew\WEB-INF\web.xml`

For more information on configuring the web.xml file for Servlet Application Explorer, see the *BEA WebLogic ERP Adapter Installation and Configuration* documentation.

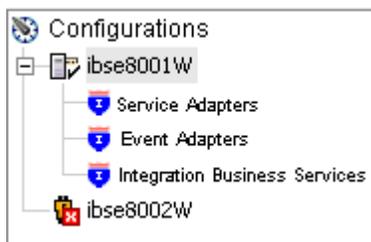
To configure ports and channels in a clustered environment:

1. Open Swing Application Explorer in BEA WebLogic Workshop.
2. Create a new connection to the iBSE instance. For information on creating a new configuration, see *Appendix A, Using Application Explorer in BEA WebLogic Workshop to Create XML Schemas and Web Services*.



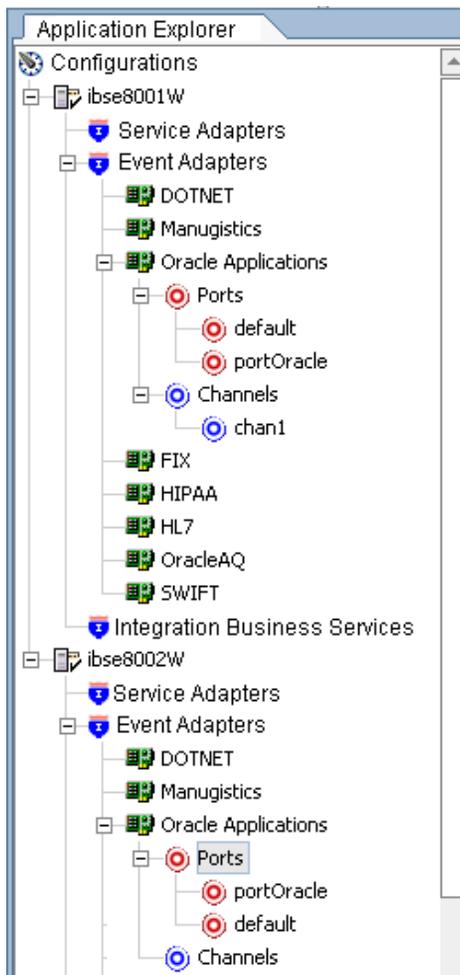
Note: Use the IP address or machine name in the URL; do not use localhost.

3. Connect to the new configuration and select the *Event Adapters* node in the left pane of Application Explorer.



4. Add a new port for the Oracle E-Business Suite adapter. For more information, see *Creating an Event Port* on page B-3.
5. Create a channel and add the port you created. For more information, see *Creating a Channel* on page B-23.
6. Click *Next* and enter the application server parameters.
7. Start the channel.
8. Create a new configuration and connect to the second iBSE instance.

The connection to iBSE must be configured to each instance of the managed server. The following image shows two configurations.



The following operations performed on one managed server will be replicated on all other managed servers:

- Create port and channel: Creates the channel and port under all available servers.
- Delete port and channel. Deletes the port and channel under all available servers.

The following operations must be performed on each server:

- Start channel. Starts the channel for the specific server.
- Stop channel. Stops the channel for the specific server.

APPENDIX C

Open Interfaces for Oracle Release Apps 11i

Topics:

- Oracle Financial
- Oracle Manufacturing and Distribution

The Oracle Application environment consists of different functional components. This section describes the agents that constitute Oracle functional components.

The following topics consist of Oracle Published Interface tables available in release 11.5.5 of Oracle E-Business Suite. The tables are grouped according to the functional modules to which they belong.

Oracle Financial

The following tables are inbound tables. Inbound tables are typically used by services and listeners. The first column of the following five tables lists the interface name, for example, Journal Entries. The remaining columns list the direction of the data flow, whether a table, view, or process, and the name of the interface table.

1 Oracle General Ledger

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
Journal Entries	Inbound	Table	GL_INTERFACE
Budget Amounts	Inbound	Table	BUDGET_INTERFACE
Daily Rates	Inbound	Table	GL_DAILY_RATES_INTERFACE

2 Oracle Accounts Payable

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
Vendor Invoices (from Invoice Gateway, EDI Gateway, and other sources)	Inbound	Table	AP_INVOICES_INTERFACE AP_INVOICE_LINES_INTERFACE
CreditCardTransactions	Inbound	Table	AP_EXPENSE_FEED_LINES

3 Oracle Cash Management

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
BankStatements	Inbound	Table	CE_STATEMENT_HEADERS_INT_ALL CE_STATEMENT_LINES_INTERFACE

4 Oracle Receivables

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
Customers	Inbound	Table	RA_CUSTOMER_INTERFACE RA_CUSTOMER_PROFILES_INTERFACE RA_CONTACT_PHONES_INTERFACE RA_CUSTOMER_BANKS_INTERFACE
CustomerInvoices	Inbound	Table	RA_INTERFACE_LINES RA_INTERFACE_SALESCREDITS RA_INTERFACE_DISTRIBUTIONS
ReceiptsfromBanks	Inbound	Table	AR_PAYMENTS_INTERFACE_ALL

5 Oracle Fixed Assets

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
CreateAssets (mass additions)	Inbound	Table	FA_MASS_ADDITIONS

Oracle Manufacturing and Distribution

The following tables are either inbound or outbound tables. Inbound tables are typically used by services and listeners. Outbound tables are typically used in conjunction with the SQL Listener. The first column of the following eight tables lists the interface name, for example, Transactions. The remaining columns list the direction of the data flow, whether a table, view, or process, and the name of the interface table.

6 Oracle Inventory

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
Transactions	Inbound	Table	MTL_TRANSACTIONS_INTERFACE MTL_SERIAL_NUMBERS_INTERFACE MTL_TRANSACTION_LOTS_INTERFAC E

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
DemandInterface	Inbound	Table	MTL_DEMAND_INTERFACE
OnHandBalances	Outbound	View	MTL_ITEM_QUANTITIES_VIEW
UserDefined Supply	Inbound	Table	MTL_USER_SUPPLY
UserDefined Demand	Inbound	Table	MTL_USER_DEMAND
Replenishment	Inbound	Table	MTL_REPLENISH_HEADERS_INT MTL_REPLENISH_LINES_INT
Item	Inbound	Table	MTL_SYSTEM_ITEMS_INTERFACE MTL_ITEMS_REVISIONS_INTERFACE
CustomerItem	Inbound	Table	MTL_CI_INTERFACE
Customer Item CrossReferences	Inbound	Table	MTL_CI_XREFS_INTERFACE

7 Oracle Engineering / Oracle Bills of Material

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
MFGCalendar	Outbound	View	BOM_CALENDAR_MONTHS_VIEW
BillOfMaterial	Inbound	Table	BOM_BILL_OF_MTLS_INTERFACE BOM_INVENTORY_COMPS_INTERFACE BOM_REF_DESGS_INTERFACE BOM_SUB_COMPS_INTERFACE MTL_ITEMS_REVISIONS_INTERFACE
Routings	Inbound	Table	BOM_OP_ROUTINGS_INTERFACE BOM_OP_SEQUENCES_INTERFACE BOM_OP_RESOURCES_INTERFACE MTL_RTG_ITEM_REVS_INTERFACE

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
ECO (engineering change orders)	Inbound	Table	ENG_ENG_CHANGES_INTERFACE ENG_ECO_REVISIONS_INTERFACE ENG_REVISIED_ITEMS_INTERFACE BOM_INVENTORY_COMPS_INTERFACE BOM_REF_DESGS_INTERFACE BOM_SUB_COMPS_INTERFACE

8 Oracle Cost Management

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
ItemCostInquiry	Outbound	View	CST_INQUIRY_TYPES CSTFQVIC (View Item Cost Information)
MFGCostReporting	Outbound	View	CST_REPORT_TYPES CSTRFICR (Inventory Valuation Report)
PeriodicCost (only for the first opened period)	Inbound	Table	CST_PC_ITEM_COST_INTERFACE CST_PC_COST_DET_INTERFACE

9 Oracle Master Scheduling/MRP and Oracle Supply Chain Planning

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
Forecast	Inbound	Table	MRP_FORECAST_INTERFACE
ForecastEntries	Inbound	PL/SQL Table	T_FORECAST_DESIGNATOR
MasterSchedule	Inbound	Table	MRP_SCHEDULE_INTERFACE
MasterScheduleRelief	Inbound	Table	MRP_RELIEF_INTERFACE

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
Planner Workbench	Outbound	Process	Stored Procedure MRPL06, or WIP_JOB_SCHEDULE_INTERFACE PO_REQUISITIONS_INTERFACE PO_RESCHEDULE_INTERFACE
Projected Requirements	Outbound	Table	MRP_RECOMMENDATIONS
Projected Supply	Outbound	Table	MRP_GROSS_REQUIREMENTS

10 Oracle Order Management/Shipping

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
OrderImport	Inbound	Table	SO_HEADERS_INTERFACE_ALL SO_HEADER_ATTRIBUTES_INTERFACE SO_LINES_INTERFACE_ALL SO_LINE_ATTRIBUTES_INTERFACE SO_LINE_DETAILS_INTERFACE SO_SALES_CREDITS_INTERFACE SO_PRICE_ADJUSTMENTS_INTERFACE
DeliveryBased Ship ConfirmOpen Interface	Inbound	Table	WSH_DELIVERIES_INTERFACE WSH_PACKED_CONTAINER_INTERFACE WSH_PICKING_DETAILS_INTERFACE WSH_FREIGHT_CHARGES_INTERFACE

11 Oracle Purchasing

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
Requisitions	Inbound	Table	PO_REQUISITIONS_INTERFACE PO_REQ_DIST_INTERFACE

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
Requisition Reschedule	Inbound	Table	PO_RESCHEDULE_INTERFACE
Purchasing Documents	Inbound	Table	PO_HEADERS_INTERFACE PO_LINES_INTERFACE
Receiving	Inbound	Table	RCV_HEADERS_INTERFACE RCV_TRANSACTIONS_INTERFACE

12 Oracle Quality

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
CollectionImport	Inbound	Table	QA_RESULTS_INTERFACE
DynamicCollection PlanView	Outbound	View	Q_COLLECTION_PLAN_NAME_V
DynamicCollection ImportView	Inbound	View	Q_COLLECTION_PLAN_NAME_IV

13 Oracle Work in Process

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
WIPMoveAgent	Inbound	Table	WIP_MOVE_TXN_INTERFACE
Resource	Inbound	Table	WIP_COST_TXN_INTERFACE
WIPWorkOrder Agent	Inbound	Table	WIP_JOB_SCHEDULE_INTERFACE
Material	Inbound	Table	MTL_TRANSACTIONS_INTERFACE MTL_SERIAL_NUMBERS_INTERFACE MTL_TRANSACTION_LOTS_INTERFACE

Interface Name	Data Flow Direction	Table, View, or Process	Interface Table Name
WIP Scheduling	Outbound/ Inbound	Processes and Table	Stored Procedures LOAD_INTERFACE and LOAD_WIP and WIP_SCHEDULING_INTERFACE

APPENDIX D

XML Business Functionality in the Adapter

Topics:

- BEA WebLogic Adapter for Oracle E-Business Suite and Oracle Functionality
- XML Business Functionality Descriptions

This section describes the business functionality of the XML supported by the BEA WebLogic Adapter for Oracle E-Business Suite.

BEA WebLogic Adapter for Oracle E-Business Suite and Oracle Functionality

The BEA WebLogic Adapter for Oracle E-Business Suite supports Oracle Advanced Queuing for services and events and for iWay Business Services Engine (iBSE) Web services. This provides direct bidirectional access to the Oracle Internet Procurement Connector for direct Open Application Group (OAG) formatted XML and also to Oracle XML Gateway, which supports all DTD-based XML standards. The majority of Oracle prebuilt messages delivered with Oracle E-Business Suite are pre-mapped using the Open Application Group (OAG) standard. Any Oracle prebuilt message may be re-mapped to any standard of choice using XML Gateway Message Designer.

Oracle Internet Procurement (OIP) Connector provides for easy integration between the Oracle Internet Procurement application and external systems. OIP Connector is based on the open, industry-standard Open Application Group Internet Standard (OAGIS) XML. For more information on OAGIS XML, see the Open Application Group Web site at <http://www.openapplications.org>. The standard allows OIP Connector interaction with middleware systems adhering to the OAGIS standard.

Oracle XML Gateway is a set of services that allows for easy integration with Oracle E-Business Suite to support non-standard XML messaging. Oracle E-Business Suite utilizes the Oracle Workflow Business Event System to publish and subscribe to application business events to automatically trigger the creation and consumption of XML messages.

Oracle XML Gateway consumes events raised by Oracle E-Business Suite, and subscribes to inbound events for processing. Oracle XML Gateway uses the message propagation feature of Oracle Advanced Queuing to integrate with the Transport Agent to deliver or receive messages to and from business partners.

XML Business Functionality Descriptions

This topic describes the business functionality of the XML supported by the BEA WebLogic Adapter for Oracle E-Business Suite.

Post Journal

Journal is an XML business document that transmits data necessary to create a journal entry from any sub ledger business application to a general ledger application. This scenario assumes that the details of the financial transactions are kept in the sub ledgers and that the drill back mechanism from the general ledger component to the sub ledger components is addressed by this XML. Many applications in the enterprise environment create data that causes changes in the account balances of a general ledger application (such as Benefits, Costing, Human Resources and Payroll, Inventory, Manufacturing, Production, and Treasury). By no means is this a complete list of all the activities which may generate a journal entry. Many other tasks that occur within the enterprise applications cause the creation of a General Ledger journal entry. For example, the adjustment of inventory value is a task that occurs within Inventory.

Journal supports the summarization of accounting activities from sub ledgers to the general ledger. This summary is used to express account balances, which are used to report on the financial condition of the enterprise. A Journal XML consists of the Journal Entry Header information which includes the type of journal being entered, a Journal Entry Detail Line (typically, there are at least two occurrences given the common accounting rule of every debit requiring an equally balancing credit), and an account code, which consists of a number of separate elements. Some of these elements can be linked to specified XML fields; for example, the nominal account key (GLNOMACCT) or the cost center (COSTCENTER).

Confirm

The purpose of the Confirm XML is to add a layer of software application to the software application exception capability to the integration solution. The Confirm XML does this by providing a mechanism for the business software layers to communicate to each other in addition to the confirmation mechanisms that may be provided with any middleware tools involved in the solution.

In instances when the receiving application must communicate its native informational messages to an integrated partner application, it is important to ensure that the applications have a mechanism for speaking their own language to each other.

The use of the Confirm XML is monitored using a confirmation indicator set by the sending business software application. This is accomplished by setting the Confirmation Indicator (CONFIRMATION) in the Control Area of the originating XML. The Confirmation Indicators are defined as follows:

- 0 = Do not send back a confirmation XML.
- 1 = Send back a confirmation XML only if an error has occurred.
- 2 = Send a confirmation XML regardless of whether an error has occurred.

Confirm XML:

1. Ensures that the message was received and understood within the application or component
2. Communicates error conditions at the application level (such as when a field is missing, a customer does not exist, and so on).
3. Communicates that a critical update was successful (or unsuccessful), such as an update to inventory, credit balance, or ledger balance.

Process PO

A Purchase Order is an XML business document that an organization issues to request delivery of goods or services for specific dates and locations.

The Process PO XML is used to transmit a purchase order to a supplier's order management application. The Process PO is the task of sending the electronic form of a purchase order document to a supplier. This is designed as an external purchase order.

Data such as name, currency, payment method, bill-to and ship-to addresses, and payment terms, along with information relevant to the supplier, is transmitted to the supplier to engage an order for goods and services.

Acknowledge PO

The Acknowledge PO XML is a document used to acknowledge receipt of the Purchase Order and to reflect any changes. This is designed as an external purchase order.

Commonly, the acknowledgment is generated by an order management application and transmitted to a purchasing or procurement application. Acknowledgement (when the entire document is accepted, rejected, or modified) represents feedback from the supplier concerning the original purchase order received.

Load Receivable

Receivable is an XML business document that transmits data to create a receivable open item in Oracle receivables from the billing information generated in an order management application. Receivable may also update the general ledger, depending on the specific architecture of the accounting application.

The scope of receivables is to create an XML to recognize customer obligations. Specific transactions include:

- Sales Invoice
- Credit Memo
- Debit Memo
- Charge Back

Receivable XML may also be used for transactions that do not originate from an order management application.

The receivable application may be a direct sub-ledger of the general ledger. Updates to G/L balances occur using the receivable module; therefore, the receivable XML contains both receivable and general ledger transaction information.

The other environment may also exist when general ledger updates occur directly from the Order Management application. The reconciliation between the receivable and general ledger is a function within the financial applications rather than of the integration space. This model allows the G/L balances to be updated in either detail or summarized form.

The role of the receivable application includes functions such as:

- Allowing Cash Application
- Dunning
- Dispute management

Relevant data that identifies the receivable includes header information (totals, identifier), partner location and contacts, invoice payment terms, and tax information.

Load Payable

Payables is an XML business document that transmits data from the purchasing information generated in a purchasing application to create a payable open item in a payables application. This may also update the general ledger, depending on the specific architecture of financial applications.

Payables indicates that the supplier's invoice is ready to be paid and has been approved before the information moves to the accounts payable application. An approved invoice is also known as a voucher. The application later defines invoices that are matched within the accounts payable application in a separate business service request.

Some financial applications have the general ledger and accounts payable databases tightly integrated where updates to the accounts payable application are automatically reflected in general ledger balances. Payables transmits all information needed for both the accounts payable and the general ledger. Other applications allow the general ledger balances to be updated separately from the accounts payable. In this case, the payables and journals XML accomplishes this scenario.

Sync Customer

The purpose of the Sync Customer XML is to keep customer information that exists on separate databases synchronized. The Sync Customer XML allows the adding of new customers and the modification of previously established customers. These customer records are used to reference invoices in a billing system.

Sync Supplier

The Sync Supplier XML facilitates keeping supplier information that exists on separate data bases synchronized. The Sync Supplier allows the adding of new suppliers and the modification of previously established suppliers.

This record consists of relevant data to a partner (supplier name and address, contact information associated with the partner, and references of supporting documents).

Load LdgrBudget

The LDGRBUDGET is an XML business document that transmits budget amounts from all possible source applications throughout an enterprise to a general ledger or budget application. Usually, budget data is created using a spreadsheet by each organization in an enterprise. Once budget data is ready, LDGRBUDGET XML facilitates the transfer of budget amounts against a revenue or expense account codes to a general ledger application for controlling expenditures and creating variance and analysis reports.

Get PO and Show PO

The Get PO is an XML business service request that enables a business application module to request information concerning a specific purchase order from Oracle Purchasing. The reply to this request is the Show PO XML business request. There are a variety of business applications in several environments that may use this capability. For example, an MRP application may use this capability to ask for information from Oracle Purchasing, or a Plant Data Collection application may also use it to request information from Oracle Purchasing. This XML does not usually cause updates to occur.

As a Get PO request is made, the Show PO XML supplies purchase order information to another business application module. This request is also used as push notification of an event. There are many possible business applications in several environments that use this capability. For example:

1. Oracle Purchasing could use this request to send information to a Plant Data Collection application.
2. Other modules such as MRP, Inventory, or Manufacturing could use this to obtain order information.
3. The PO application can notify the MRP/Inventory application when a vendor gives or changes a promise to deliver.

Receive PO

The Receive PO is an XML business service request that supplies the information that the Oracle Purchasing module requires to assign a receipt posting tag to a purchase order. Oracle Purchasing uses the receiving and inspection information supplied by the Receive PO XML to ensure that the organization only accepts and pays for the items ordered, received, or inspected, depending on the identified matching rule. A variety of business applications in several scenarios may use this capability. For example:

- Receipt of information from other non-Oracle systems.
- Receipt of barcoded and other receiving information from scanners.
- Advance Shipment Notices (ASNs) sent from suppliers.

As the Receive PO XML supplies the information, Oracle's Receiving Open Interface maintains the integrity of the new data as well as the receipt data already in Oracle Purchasing.

Get Prodorder

Prodorder is an XML business document that details the work order information. This document may be related to three business service requests: Get Prodorder XML, Show Prodorder XML and Receive Prodorder XML. This enables the organization to import discrete job and repetitive schedule information, discrete job operations, and material, resource and scheduling information from any source.

Show Prodorder and Receive Prodorder

The Get ProdOrder XML enables a business application module to request specific work order information from another business application module. Sources may include handheld devices, other manufacturing execution systems, planning systems, and order entry systems. The reply to this is the SHOW PRODORDER XML. Information on the following items may be used to request a work order: the pre-determined bill of material structure, lot, or serial information about the item; the operation in the routing at which to change the work order; and the accounting information that can optionally accompany a change in the order.

The Show ProdOrder is an XML business document that supplies work order information to another business application module. The environment for this request can be within the enterprise or outside the enterprise. On the other hand, the Receive Prodorder XML is a business service request that supplies information which the ERP system requires to do receipt posting against a work order.

Transfer Item

The Transfer Item XML is a business service request that enables organizations to do sub-inventory or direct inter-organization transfers in Oracle inventory. As a company defines multiple inventories, warehouses, and manufacturing facilities as distinct organizations, the Transfer Item XML allows for the efficient performance of transfer of one or more items in a single transaction. This also allows the transfer of partial quantities of the same item to different sub-inventories and locations.

Get Issueinfo and Show_IssueInfo

IssueInfo is an XML business service request for information against an order, from the ERP system into a plant data collection system to confirm the item issue transaction. The environment for this integration is from plant data collection systems to the Oracle manufacturing modules. Information concerning the work order to which the material item is being issued, the lot or serial number for the work order item, and the operation in the routing of the production item is transmitted to request an item issue.

The purpose of the Show IssueInfo XML business service request is to supply to another system item issue information against an order to confirm the issue transaction against that order. Examples of order types would be a work order, a sales order, a service order, or a maintenance order. This XML may be used individually, or as part of a larger interface scenario. Oracle Manufacturing (including Inventory and MRP) - to interface with Oracle or non-oracle Order entry systems, work in process systems, and the plant maintenance systems using this IssueInfo XML - allows for the issuance against an order.

Confirm Issue

Issue is an XML business service request used to notify Oracle Manufacturing of the issue of required material to a work order for making a product. This XML is also used to notify Oracle Manufacturing of the return of material from a work order or job order back into inventory. The business environments most likely to require this capability are any type of manufacturing scenario.

The Issue XML communicates what item is being issued, where it is being issued from, which processing operation it is being issued to, what quantity was issued, and at what time this event occurred. In the case of a return, this communicates what item is being returned, which processing operation it is being returned from, which sub-inventory location it is being returned to, the quantity being returned, and the time at which this event occurred. This XML commonly causes updates to occur.

Issue MiscItem and Receive MiscItem

The MiscItem XML is another business service request that reflects an unplanned issue or receipt of an item to or from a miscellaneous location. Possible reasons for issuance include somebody broke the material, or the material is defective and needs replacing, or the material is used up and needs replenishment. Miscellaneous transactions may be issued from a plant data collection system or an inventory system to Order Management or Manufacturing modules. The MiscItem XML may be used to issue materials to groups that are not inventory, receiving or work in process such as research and development group. This can also assist in the issuance of items to individuals or projects or to issue damaged items to expense accounts such as scrap. The MiscItem XML may also be used to notify a corresponding business application to reflect an unplanned receipt of an item to a miscellaneous location. This is primarily designed for supplies items or MRO items.

Get Countinfo and Show Countinfo

Countinfo is an XML business service request to enable a business application to request and show inventory on-hand quantities from or to an Oracle Inventory system. The Countinfo XML enables a business application to request on-hand quantity information from an ERP system. This may also be used to show inventory count information to enable Oracle Inventory to send inventory count information to a PDC or other application system. This request may be used as a response to a get countinfo request or as a push notification of an event. There are many possible business applications in several environments that may use this capability.

Update Invencount

Invencount is an XML business service request that transmits an inventory count to Oracle Inventory from the actual physical inventory location. This count may be a cycle count or a physical count. This XML may also apply to planned or unplanned inventory counts. The Invencount XML allows the user to transmit physical inventory count information that can be used to reconcile system-maintained item on-hand balances with the actual counts of inventory. Accurate system on-hand quantities are essential for managing supply and demand, maintaining high service levels, and planning production.

Get Credit, Show Credit, and Update_credit

Credit is an XML business service request used to either Request, Show, or Update credit. This also includes Change Status request used to keep order, shipment and open items amounts current. This contains all of the information necessary to make a decision to give credit to a customer. The Request is made for the Order Management to request credit data for a customer from Oracle Receivables. This does not imply any update, it is only an inquiry function. The Show Credit XML is used as a response back to the order management application. The Update Credit XML may be used in both directions between the order management and the accounts receivable application. Its purpose is to keep order, shipment and open item amounts current. The Update Credit XML also transmits changes in the accounts receivable open item balances to the credit management function of the customer order management application.

Change Status

The Change Status XML request is used to update the order management application with any changes in business status for a particular customer. The purpose of this request is to notify the customer order management application that the overall credit status of a customer has changed or status on specific order(s) are to be changed.

Load Projacctg

Projacctg is an XML business service request that enables all relevant sub-systems (such as Accounts Payable for payment of materials used, Accounts Receivable for invoicing of billing projects, Budget for validation, Purchasing to report committed costs, Assets to record capitalized projects and Human Resources system to update employee information) that submit single sided transactions to send information to Oracle Project Accounting. Applications which produce double-sided transactions would use other XML business service requests to update the project accounting application. For example, order management or purchasing applications would use the Load Receivable or Load Payable XML. The Projacctg XML assists in providing project-oriented companies with a flexible approach to define and structure projects, tasks and budgets by which to monitor project status. Integration with other applications allows effective accounting for costs or to process revenue and invoices.

Sync Projinfo

The purpose of the Sync Projinfo XML business service request is to enable all relevant sub-systems that submit transactions to Oracle Project Accounting to maintain valid values for the key project fields. The target applications for this update would include, but not necessarily be limited to: Accounts Payable, Accounts Receivable, Budget, Order Management, Purchasing, Time and Labor, Travel and Expense.

Get Wipconfirm and Show Wipconfirm

Get WIP Confirm is an XML business service request that enables the requesting of data necessary to perform a confirmation of the movement of WIP (Work in Process). This does not commonly cause updates to occur. This template may be used individually, or as part of a larger interface scenario with a plant data collection system or a shop floor control system. Relevant data to get confirmation of the movement of WIP include: information concerning the specific work order or job order and the WIP operation or step in a routing from which the product is being completed into inventory or at which to return back into manufacturing, accounting information collected in the WIP Completion and Return transaction, information concerning the production order in the transaction as reference to an existing job order, and information concerning the resources associated with a particular WIP operation or sub-operation within a routing.

As a response to the Get WIP Confirm XML, the XML template Show WIP Confirm is used.

Update Wipconfirm

Update WIP Confirm is an XML business service request that notifies Oracle Manufacturing of the completion of an end product in the production process and movement of that product to a finished goods inventory. This XML also notifies a Oracle Manufacturing of the return of end product from a finished goods inventory back into the production process. The business environments most likely to require this capability are any type of manufacturing scenario. This communicates which processing step the product is coming from, the quantity moving, the inventory organization or sub-inventory location it is moving to, and the time at which this event occurred. In the case of a return, the request communicates which inventory location the product is coming from, the quantity moving, the processing step it is moving to, and the time at which this event occurred.

Get PickList

The Get PickList is an XML business service request that enables a request for the retrieval of a single Pick Slip from an ERP system. A pick slip or pick list is an internal shipping document pickers use to locate items to ship for an order. The reply to this request is the Show PickList. Individual lines from a Pick Slip are not selectable with this request. Only the complete document is selected and returned. Information relevant to the request include: a picking document that is generated in an ERP shipping module or Oracle Order Management.

List PickList

List Picklist is another XML business service request that provides a list of Pick Slips from an ERP system to another application. This may be initiated in response to a GETLIST PICKLIST request or upon some business event. When a receiving application receives this XML, the information can be used as is or it may be used to initiate a selection of a specific Pick slip through the GET PICKLIST request. The processing is designed to provide multiple occurrences of summary data. This does not usually cause updates to occur.

The LIST XML is used for the receiver of the GETLIST XML to respond with the results of the search that was initiated by the GETLIST. Each of the records being returned is contained within an instance of the LIST_PICKLIST element underneath the DATAAREA Element.

The attributes associated with the LIST XML are as follows:

- `rsstart` attribute is a number that indicates the starting record for the section of the resulting set returned in the list message. This value should always match the `rsstart` value in the originating GetList XML.
- `rscount` attribute is a number that indicates the number of records returned in the message. The subsequent request for additional records should have a `rsstart` value of `rscount + 1`.
- `rstotal` attribute is a number that indicates the total number of records in the result set.
- `rscomplete` attribute is a Boolean that indicates that the list provided exhaust the possible values.
- `rsref` attribute is a string that represents the implementation-specific result set identifier for subsequent requests.

Show PickList

The purpose of the SHOW PICKLIST XML is to show the details of an individual Pick Slip from an ERP system. This may be sent in response to a GET PICKLIST or it may be initiated upon some business event. This does not usually cause updates to occur. The picking document, the lines on a specific picking document, and details about a line item on a Pick Slip (such as date and time of loading and shipping) that are generated in an ERP shipping or order management system are some of the data that appears on the request.

Update PickList

Update Picklist is an XML business service request that updates the details of an individual Picking List from a plant level to an ERP system. A pick slip or pick list is an internal shipping document pickers use to locate items to ship for an order. This usually causes updates to occur.

Get Personnel and Show Personnel

Personnel is an XML business service request that may be used to either Get Personnel to request personnel data for a resource or Show Personnel to provide personnel data for a resource to a requesting business application. This facilitates integration between a human resource system to manufacturing systems such as shop floor and plant data collection. Personnel information such as employee name, cost center, division, employee category, status, job code, shift and wage group are required to manufacture, cost and schedule products.

Update Persontime

The UPDATE PERSONTIME is an XML business service request that updates work time information for an employee from a data collection application to an ERP Human Resource application. This causes updates to occur. This may pass on data to update personnel information, employee category, employee status, overtime, and the quantity of employees' reporting hours, days, and so on.

Sync Field

Sync Field is another XML business service request that enables the validation of data that exists on separate application's databases. This request can cause on-line validation to occur or may be a single tool for synchronizing data. In Oracle Financial, for example, this is very useful when a company engages in consolidation or inter-company transactions of which each balancing company resides in separate databases.

Sync Personnel

The XML called Sync Personnel is a business service request that enables the synchronization of employee data that exists on separate databases between manufacturing and human resource applications. The Sync Personnel allows the adding of new employees and their relevant data as well as the modification of previously established employees. The Sync Personnel is used to facilitate the maintenance of human resource data in a manufacturing work force planning module, in Oracle this may be under Master Scheduling module. This enables the workforce planning module to use current personnel information when creating finite production schedules. This can also be used by Oracle Project Accounting or a work order management application to assign qualified personnel or to perform resource planning. Employee data such as employee name, category, job code, position, shift and competency are ut a few data that need to be synchronized between a manufacturing and a human resource application.

Sync Wrkschedule

Sync Wrkschedule is an XML business service request that enables the synchronization of Work Schedule data that exists on separate databases. The Sync Wrkschedule allows the adding of new Work Schedules as well as the modification of previously established Work Schedules. This causes updates to occur and may be used as part of a large integration scenario or as a single tool for synchronizing data. This is designed primarily to enable synchronization of data in a Human Resource to Manufacturing Application integration scenario. Personnel work schedules detailed in a human resources system may be imported or exported to or from a manufacturing application such as Oracle Manufacturing.

Sync Mfgtlcode

The **Sync Mfgtlcode** is an XML template that provides manufacturing codes that need to be captured with the labor hours in a time and labor reporting application. The Sync Mfgtlcode can also be used to provide other operational codes (such as medical codes) to a time and labor reporting application. The reason for associating these codes with labor hours is due to informational requirements of the manufacturing and/or financial systems.

The Sync Mfgtlcode may not be useful or necessary in integration scenarios in which one of the following is true:

1. The manufacturing codes observe complex hierarchical relationships or valid combinations. In this scenario, a plant data collection application or shop floor control application would probably be the labor reporting system,
2. Large volume repetitive manufacturing environments or process manufacturing in which none of the fields contained in the Sync Mfgtlcode need to be associated with labor hours by the manufacturing or financial applications.

Sync COA

COA is an XML business service request that distributes general ledger chart of accounts (COA) code identifiers to other applications to store for validation purposes. This scenario assumes that the general ledger “owns” the chart of accounts definition and the instances of data within it. The sub ledger applications have several choices for validation of account numbers and other fields in the complete chart of accounts structure. One of these choices is to synchronize the chart of accounts structure and data from the general ledger application to all of the sub ledgers. The Sync COA allows for validation of account codes and account code combinations used in the subledger to record transactions and eventually post to the account balances in general ledger. The synchronization process can ensure that all of the components in the integration scenario have the same data necessary to communicate.

Sync Exchngrate

The purpose of the **Sync Exchngrate** XML business service request is to enable the passing of updates of currency exchange rates to other applications that have exchange rate tables. In Oracle Financial for example, daily rates and period-end rates tables are maintained in Oracle GL. The sync Exchngrate XML facilitates the updating of exchange rates for other subsystems that require rates to convert or revalue foreign currency denominated transactions.

Add Requisitn

Add Requisitn is an XML business service request that sends demand for goods or services to another business application for consideration of buying or in some way obtaining the requested items. In Oracle, this facilitates the sending of a purchase requisition to trigger the issuance of a purchase order for items requested. This XML usually causes updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand. This request may originate from employees or requestor (using a self-service application), or may be triggered by Manufacturing modules such as MRP or Inventory when the reorder point reaches a level that signals the planner to issue a request for materials for production. Data such as supplier name, payment method and information that describes the requested item and its attributes including sub-components or sub-assemblies are sent to another business application (may be Oracle Purchasing) for validation and approval to purchase. The inclusion of a REQUISTNID (requisition ID) ties back a document to its origin, thus providing an identifier that enables drill back audit trail functionality.

Change Requisitn

The **Change Requisitn** XML business service request communicates changes to an existing purchase requisition for goods or services. This change must refer to the original document and/or item requested. The change processing assumes replacement of fields sent, with the exception of REQUISTNID and REQLINENUM fields. If any of the Field Identifiers above require changing, that constitutes a cancellation of the request and/or the addition of another requisition. This may usually cause updates to occur and may be used as part of a larger integration scenario (between an Inventory or MRP system to a purchasing system) or as a single tool for communicating demand.

Cancel Requisitn

The **Cancel Requisitn** XML business service request communicates from one business application to one or more other business applications that a previous requisition or requisition line item is no longer needed. This cancel must refer to the original document and/or item ordered. The user may either cancel the entire requisition or only specific lines on the requisition. To cancel the entire requisition, include only the REQUISITN Header information for the instance of the requisition you wish to cancel. To cancel a line or several lines, each line to be cancelled must be included. This XML commonly causes updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand.

Get Requisitn and Show Requisitn

An XML business service request called **Get Requisitn** enables a business application to request information concerning a specific requisition from another business application which is usually a purchasing system. The reply to this XML is the Show Requisitn XML. This Get Requisitn XML does not usually cause updates to occur. It may be used as part of a larger integration scenario or as a single tool for requesting information on existing demands for goods or services. Other users may need requisition information to validate statistics on growing demand for a specific item for planning purposes.

To send information relative to demand for goods or services to another business application or may be a notification vehicle initiated upon an event in a business application, the Show Requisitn XML business service request is used. The Show Requisitn XML usually does not cause updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand. All data relevant to a specified purchase requisition (such as Supplier information, information describing the requested items and its attributes, sub-components and sub-assemblies) is sent via the Show Requisitn XML.

Getlist Requisitn

Getlist Requisitn is an XML business service request enabling a business application to request summary information for one or more requisitions from another business application. The Getlist Requisitn also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all Requisition Lines for a specific ITEM. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. This XML does not usually cause updates to occur. It may be used as part of a larger integration scenario (between a manufacturing application and a purchasing application) or as a single tool for requesting information on existing demands for goods or services.

Many of the Data Types and the Field Identifiers contained within this XML may be used to select required information. This is done by requesting specific Field Identifiers or by requesting an entire Data Type. If the Data Type is not using Field Identifiers to select information, but the data within a Data Type is requested to be returned, the Data Type is coded without any Field Identifiers. This signifies to the responding application that all of the data that corresponds to that Data Type is to be included in the response. It is also possible to request that a range of values be returned the first occurrence of the XML is treated as the from and the second optional occurrence is treated as the to value. The maxitems attribute defined below indicates the maximum number of entries to return in the List XML that is used in response to this GETLIST.

List Requisitn

To send information relative to demand for goods and services to another business application the **List Requisitn** XML business service request may be used. This may be in response to a Getlist Requisitn request, or it may be a notification vehicle, initiated upon an event in a business application. The LIST verb describes the behavior of supplying one or several documents in a summary format to the requesting business application. These listings of information may be supplied for requisition documents, or requisition lines and/or requisition sub lines. This XML usually does not cause updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand.

The List XML is used for the receiver of the Getlist XML to respond with the results of the search that was initiated by the Getlist. Each of the records being returned is contained within an instance of the LIST_REQUISITN element underneath the DATAAREA Element.

Getlist PO

Getlist PO is an XML business service request that enables a business application to request information containing summary information for one or more Purchase Orders from another business application. The Getlist PO also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all purchase order lines for a specific item. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. This does not usually cause updates to occur. It may be used as part of a large integration scenario or as a single tool for requesting information on existing demands for goods or services. A Plant Data Collection application could use this XML to request information from Oracle Purchasing or a MRP, Inventory or Manufacturing business application could use this to obtain order information.

Many of the Data Types and the Field Identifiers contained within this XML may be used to select required information. This is done by requesting specific Field Identifiers or by requesting an entire Data Type. If the Data Type is not using Field Identifiers to select information, but the data within a Data Type is requested to be returned, the Data Type is coded without any Field Identifiers. This signifies to the responding application that all of the data that corresponds to that Data Type is to be included in the response. It is also possible to request that a range of values be returned the first occurrence of the XML Element is treated as the from and the second optional occurrence is treated as the to value. The maxitems attribute defined below indicates the maximum number of entries to return in the List XML that is used in response to this GETLIST.

List PO

A **List PO** XML business service request is used to send information relative to demand for goods or services to another business application. This may be in response to a Getlist PO request, or it may be a notification vehicle, initiated upon an event in a business application. The LIST verb describes the behavior of supplying one or several documents in a summary format to the requesting business application. These listings of information may be supplied for Purchase Orders, PO Lines, or PO Sub-Lines. This usually does not cause updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand. There are many possible business applications in several environments that may use this capability. For example, a purchasing application could use this XML to send information to a Plant Data Collection application or a MRP, Inventory or Manufacturing business application could use this to obtain order information. The environment for this XML can be within the enterprise or outside the enterprise.

Add PO

The purpose of the XML called **Add PO** is to communicate from one business application to one or more other business applications that a Purchase Order has been added or needs to be added, depending on the business case. The environment for this XML can be within the enterprise or outside the enterprise. There are many possible business applications in several environments that may use this capability. One example of a business integration scenario where this XML could be useful is to communicate back to an Inventory system that a purchase order has been processed and added in the purchasing system. The planner or buyer may use the purchase order information (such as payment terms, miscellaneous charges, the description and price of items ordered, and the dates and quantities for delivery or shipment of ordered products) to analyze and schedule production.

Change PO

The purpose of the **Change PO** XML is to request another business application module such as Oracle Purchasing to make changes to an existing Purchase Order. This change must refer to the original document and/or item ordered. The change processing assumes replacement of fields sent, with the exception of the POID and the POLINENUM fields. If any of the Field Identifiers require changing, that constitutes a cancellation of the request and/or the addition of another Purchase Order.

Cancel PO

The **Cancel PO** is an XML business service request that allows a requestor or planner to communicate from one business application to one or more other business applications that a previous Purchase Order or Purchase Order line is no longer needed. This cancel must refer to the original document and/or item ordered. A planner or requestor may invoke this XML to change the entire purchase order or to change specific purchase order lines. To cancel the entire order, include only the Purchase Order Header information for the instance of the Purchase Order you wish to cancel. To cancel a line or several lines, each line to be cancelled must be included. This XML commonly causes updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand. In Oracle Purchasing, a cancellation of a purchase order eventually cancels accounting entries related to the order.

Update Delivery

Delivery of goods from a supplier business partner via the services of a transportation provider (carrier) is an important event in a manufacturing or a purchasing application. As goods are delivered, Purchasing controls the items ordered through receiving, inspection, transfer, and internal delivery. These features control the quantity, quality, and internal delivery of the items delivered. The **Update Delivery** is an XML business service request that enables the update of delivery information for goods or services to another business application module. This XML can be initiated by a business application based on some event and sent to one or more relevant business applications. Possible business applications in several environments that may use this capability include: a Purchasing application to notify an Accounts Payable application of a specific delivery enabling the Accounts Payable application to accurately calculate the amount it needs to pay a business partner, a PO application could use this XML to send information to a Plant Data Collection application, a Purchasing application could use this to notify a MRP, Inventory, or Manufacturing business application that a delivery has occurred and the goods are available for use or inspection, and so on.

Update Inspection

An **Update Inspection** XML business service request supplies inspection information for goods or services to another business application module. This may be initiated by the sending system upon some event occurring. There are many possible business applications in several environments that may use this capability. Examples include: a PO application could use this to send information to a Plant Data Collection application, or vice versa; a MRP, Inventory, Purchasing or Manufacturing business application could use this to communicate inspection information; a Laboratory Information System could send quality information to an Inventory application; or a Quality Control application could send information to a MRP, Inventory, or Purchasing application. Matching of supplier invoices against a purchase order may be defined in Oracle Purchasing or Oracle Payables at a four-way level, that is price and quantity billed must match quantity received (or delivered), ordered and inspected. The information supplied by the Update Inspection XML supports the four-way matching in Oracle. The XML also describes reasons for quantities rejected in the INSPECTION Data Type.

Sync Item

Sync Item is an XML document that supplies information for goods or services to another business application module. This XML may also be initiated by the sending system upon some event occurring. This XML is not for synchronizing ITEM quantities at each inventory location. The Sync Inventory XML is used for this purpose. There are many possible business applications in several environments that may use this capability. For example, a MRP, Inventory, or Manufacturing business application could use this to communicate item information. This XML can be used to synchronize items used in finished goods, raw materials, work-in-process or components in a bill of materials. Data such as attributes of additional units-of-measure for the item, attributes of cost or value of an item and the location the item may be kept are synchronized for a more efficient handling of the items.

Sync Sitelevel

Sync Sitelevel is an XML business service request that enables a mechanism to ensure that the physical location identifiers (physical locations of organizations or the location codes and their meanings) are synchronized between the business applications that require this to communicate clearly. This is particularly critical when only the codes that identify locations are used. Without the meaning of the codes clearly communicated, the integration is not effective. This XML enables the SITELEVEL codes to be synchronized among business applications. This may also be initiated by the sending system upon some event occurring. There are many possible business applications in several environments that may use this capability. For example, a MRP, Inventory, or Manufacturing business application could use this to communicate SITELEVEL information. The environment for this XML can be within the enterprise or outside the enterprise.

Get Prodavil

Get Prodavil is an XML business service request that enables requests of product availability data by an Order Management business application (such as Oracle Management) to an Available to Promise (ATP) or Production business application. The business process scenario is the Order Management application interacting with the Available to Promise or Production application in order to determine availability of a product for the customer. This scenario is commonly referred to as Make to Order or Build to Order. The response to this request is the Show Prodavil. In the case where Finished Goods Inventory resides with the Order Management business application, Order Management may look at its' own Finished Goods Inventory before asking the ATP for product availability. Otherwise, The Order Management application always looks at the Available to Promise. The customer implementation rules or the business applications' capabilities usually determine this processing. In the case where there is a Finished Goods Inventory application on both sides of the integration, it is assumed that the Inventory applications are synchronized. In the case where Order Management is using the Production or Manufacturing Inventory, it is possible that Order Management would look at that Inventory availability before asking the Available to Promise application for product availability. This case is not covered at this time.

This XML usually does not cause updates to occur and may be used as part of a larger integration scenario or as a single tool for requesting product available to promise information. The picture below visualizes one possible use of this XML.

Show Prodavil

The purpose of the **Show Prodavil** XML business service request is to respond to a Get Prodavil request or to initiate the passing of product availability data from a Production or Available to Promise (ATP) business application to an Order Management business application. The business process scenario is the Order Management application interacting with the Available to Promise or Production application in order to determine availability of a product for the customer. This scenario is commonly referred to as Make to Order or Build to Order. This may or may not cause updates to occur and may be used as part of a larger integration scenario or as a single tool for requesting product available to promise information. Data necessary to communicate ATP such as the item, item description, available quantity, date and time of availability is passed from a production application to an order management application.

Update ProductReq

Update Productreq is an XML business service request that enables a business application such as Order Management to reserve a particular quantity of goods or services for a specific date and time. The business process scenario is the Order Management application interacting with the Available to Promise or Production application in order to determine availability of a product for the customer. This scenario is commonly referred to as Make to Order or Build to Order. In Oracle E-Business Suite, the Available to Promise is set up in the Order Management system, triggered by data from the manufacturing systems. The Update Productreq XML accomplishes this task in a two step process within this one request:

1. The receiving business application checks to see if an item is available in sufficient quantity by a specific date and time.
2. The receiving business application then reserves that quantity of inventory for that specific date and time combination if the product is available.

If the product requested is not available, the responding application may send one of two responses: either to use a Confirm XML to confirm the denial of the request or use a Show Prodavail to communicate an alternative product availability. This may be ITEM, DATE, or QUANTITY, or a combination of these. This may also be accompanied with a message in the NOTES field Identifier stating that this is an alternative. If the product requested is available, the responding application sends a Confirm XML business service request to confirm the execution of the request.

Create Prodorder

A **Create Prodorder** XML notifies a Manufacturing Application of the need to make a product or parts in a specific quantity, for a specific need by date. The business environments most likely to require this capability are an Engineer to Order or a Configure to Order manufacturing scenario. This XML business request communicates what the product configuration is and what choices have been made from the configuration. This commonly causes updates to occur. As an order for a specific product is received in Order Management, a production order is created and an available to promise is set. The environment for this XML can be within the enterprise or outside the enterprise.

Cancel Prodreq

Cancel Prodreq is an XML business service request that communicates from one business application to one or more other business applications that a previously requested item is no longer required. This cancel must refer to the original item requested. To cancel the item(s), each item to be cancelled must be included. This XML commonly causes updates to occur. Information required to identify the previously ordered item and to reset the order information gets communicated using this XML.

Sync Inventory

The purpose of the **Sync Inventory** XML is to enable the synchronization of Inventory data that exists on separate Item Master databases. This data is not the master data that describes the attributes of the item such as dimensions, weight, or unit of measure. This is data that describes the ITEM as it exists at a specific location. The primary focus of this XML is to synchronize the quantity of an item by stocking location. This may create new Inventory records. Its purpose is to either create or update existing Inventory records.

Get BOM and Show BOM

Get BOM (Bills of Materials) is an XML business service request that enables an application to request specific Item Bill of Material information from another business application module. The response to the Get BOM is the Show BOM. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to communicate Item Bill of Material information. The environment for this XML can be within the enterprise or outside the enterprise.

The **Show BOM** XML is used to supply Item Bill of Material information to another business application module. This XML may also be initiated by the sending system upon some event occurring. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to communicate Item Bill of Material information. Data relevant to the business such as information describing the BOM structure and its contents, information describing the attributes of a specific item or option within a classification are supplied by using the Show BOM XML. An example of an option would be CD-ROM for a laptop computer. Then each of the types of CD-ROM's for the option would be a separate ITEM. An example of an option class would be memory for a laptop. The options could then be 12, 24, or 40 megabytes of RAM. Each of these options would then have separate ITEM identifiers for memory modules that makes up the appropriate amount of memory. For 40 megabytes of RAM, this could be two 16 megabyte memory modules and an 8 megabyte module, or one 16 megabyte and three 8 megabyte memory modules.

Sync BOM

Sync BOM (Bill of Materials) is an XML business service request that communicates to a business application module or system the need to initiate the creation of a Bill of Material structure. This XML may be necessary to address the Make to Order, Assemble to Order, or Mixed Mode business ordering scenarios in a Order Management to Manufacturing application integration scenario. In Oracle for example, Oracle Manufacturing and Oracle Order Management use bills of material to store lists of items that are associated with a parent item and information about how each item is related to its parent. Oracle Manufacturing supports standard, model, option class, and planning bills of material. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to communicate the requirement to synchronize a Bill of Material. The synchronization process can ensure that all of the components in the integration scenario have the same data necessary to communicate. The environment for this XML can be within the enterprise or outside the enterprise.

Allocate Activity

Allocate Activity is an XML business service request that enables the update of ACTIVITY information from a production or manufacturing application to a costing application. This is necessary for applications that are based on a Dual Cycle Accounting model. This Dual Cycle Accounting model does not capture the details of the activities that caused entries to be made in the general ledger application, but instead captures them in a separate overall costing application. For Single Cycle accounting systems, the Journals XML is used to ensure that the costing information flows from the Manufacturing Application to the Financial Application. This XML commonly causes updates to occur and may be used as part of a large integration scenario or as a single tool for updating data.

Load Matchdoc

When using Oracle E-Business Suite, the purchase order and invoice matching functionality exists in Oracle Accounts Payable. But for other application suites, the matching functionality is made in the purchasing application. The invoice matching process may include several document types, including the following: for a Two way match - Purchase Order and the Invoice; for a Three way match - Purchase Order, Invoice, and the Receipt; and for a Four way match – Purchase Order, Invoice, Receipt, and Inspection results. For the four way match, it is assumed that inspection results have been updated on the Purchase Order for visibility in matching. The **Load Matchdoc** is an XML that is used to keep invoice, purchase order, goods receipt note and inspection ticket information current.

The Load Matchdoc XML is for use both by the accounts payable application and the purchasing application in exchanging the transactions that are required to be matched. When matching takes place in the accounts payable application, the purchasing application must inform the accounts payable application of the purchasing transactions (purchase orders, goods receiving notes and inspection tickets) to which the invoice (in accounts payable) is to be matched. These integration scenarios have been developed for document matching to occur at the line level within the PO document and the Invoice document. This may be a one to one relationship, or it may be a many to one relationship from Invoice to PO or from the PO to the Invoice. Charges not associated with a specific Invoice line must be matched individually. When matching takes place in the purchasing application, the accounts payable application may have to inform the purchasing application of the supplier invoice to which purchasing transactions (purchase orders, goods receiving notes and inspection tickets) are to be matched if the invoice is initially entered into the accounts payable application. Note that in some situations, invoices are entered directly into the purchase order application or are created by the purchase order application when using evaluated receipt settlement (ERS) and in this instance, it is not necessary to perform the separate integration described under this XML.

Update Matchok and Update Matchfail

After loading matching documents and matching functionality is executed in an accounts payable application, an **Update Matchok** XML or an **Update Matchfail** XML is used to send successful matching notification or failure notification to a purchasing application. The purpose of the Update Matchfail XML is to notify the purchasing application of a matching failure such as a tolerance failure. For example in Oracle Payables, a user can match payables invoices to purchase orders to ensure that only goods ordered, received and inspected are paid. Controls and tolerances are set to specify the range of variance allowable if the amounts or quantities on the invoice are greater than the amounts or quantities on the purchase order or receipt. Oracle Payables then creates distributions and checks that the match is within the defined tolerance. Oracle Payables may use the Update Matchok XML or the Update Matchfail XML to notify the purchasing application. A purchasing application may then use this data to close or adjust the purchase order records or inform the supplier of discrepancies accordingly.

Load_PLInvoice and Load Payable

Load PLInvoice XML is a business document that transmits data to create an unapproved open item in either a payables application or a purchasing application. The scope of the Load PLInvoice (load purchase ledger invoice) indicates that the supplier's invoice has not yet been approved and the invoice is to be used as part of the invoice matching process. For invoices that are approved for payment, this is handled by a separate XML business service request called **Load Payable XML**. If the matching functionality exists in a purchasing application, the Load PLInvoice XML is used to transmit data into the purchasing application, and only after matching does the PO application use the Load Payable XML to post approved invoices into the payables application. On the other hand, if the invoice matching functionality exists in the accounts payable application, the invoice is entered into accounts payable using the Load PLInvoice XML, and purchasing publishes matching document information to which accounts payable subscribes.

Get Matchdoc and Show Matchdoc

A **Get Matchdoc** XML business service request is used to enable both the accounts payable application and the purchasing application to request the transactions that are required to be matched or request information concerning matching. In both cases, the receiving application uses the **Show Matchdoc** XML to return the requested information. This XML does not usually cause updates to occur. In certain application suites, purchase order and invoice matching functionality exists in the purchasing application, while in other suites this functionality exists in the accounts payable application. If the invoice matching functionality exists in the purchasing application, the invoice is entered into accounts payable, purchasing requests invoice information using Get Matchdoc XML and accounts payable provides invoice information using a Show Matchdoc XML. If the invoice matching functionality exists in the accounts payable application, the invoice is entered into accounts payable, purchasing requests matching document information using Get Matchdoc XML and accounts payable provides matching document information using Show Matchdoc XML.

Getlist Picklist

The purpose of the Getlist Picklist XML business service request is to enable a business application to request summary information for one or more Picking Slips from an ERP system. If a list of documents is requested, that list is used in order for a selection and GET request of a specific picking list to be made, if necessary. This XML does not usually cause updates to occur. For example, this XML may be used by a Plant Data Collection system to request for summary information for a Pick slip from an Order Management system. Requestor may give specific field identifiers or request an entire data type, depending on the need for information.

Update Inventory

Update Inventory is an XML business service request that enables the update of Inventory data that exists on separate Item Master databases. This data is not the master data that describes the attributes of the item such as dimensions, weight, or unit of measure. This is data that describes the ITEM as it exists at a specific location. The Update Inventory XML assumes that one Inventory Application is the “owner” of the data. Either the owner of the data or the non-owner of the data may initiate this XML. The Inventory Application which is the primary owner of the data is determined by which one updates the financial records to be posted to the general ledger. The non-owner inventory does not update the financial records. To do so may result in duplicate postings or other errors in the financial record keeping. All Inventory events which may affect the financial records need to be sent to the owner inventory either through the Update Inventory XML or the Sync Inventory XML. This does not create new Inventory records. Its purpose is to update existing Inventory records.

Getlist Countinfo and List Countinfo

A **Getlist Countinfo** XML enables a business application to request several occurrences of summary Inventory Count information from an ERP system. This may be used for cycle counting or for physical inventory counts. The response to this request is the **List Countinfo** XML. This may be used individually, or as part of a larger interface scenario such as a Plant Data Collection and an Inventory system. The effective date and time, document ID or document type and the item may be used to get count information. This XML enables range selections. This is accomplished by including two separate occurrences of a Field Identifier. The first occurrence is the “FROM” selection and the second, or duplicate occurrence of the Field Identifier is the “TO” in the range to be selected. If a second Field Identifier is not included, the selection continues until the data no longer applies to the selection or the MAXITEMS is reached. All of the inventory documents may be requested by sending a value of “ALL” in the DOCUMENTID field.

A **List Countinfo** XML enables an Inventory application system to send multiple occurrences of summary or detail inventory count information to a Plant Data Collection or other application system. Data such as item description, quantity and serial number are examples of data that may be returned in the List Countinfo XML response.

Cancel Prodorder

Cancel Prodorder is an XML business service request that notifies a Manufacturing Application of the need to cancel a previous order to make a product in a specific quantity, for a specific need by date. This XML may be used to cancel an entire Production Order, or a specific line on the production order. A cancel must refer to the original document and/or item ordered. To cancel the entire order, include only the header information for the instance of the Production Order that needs to be cancelled. To cancel a line or several lines, each line (as indicated on the predetermined Bill of Material structure) to be cancelled must be included in the request. An Order Management system may send this request to a manufacturing system (in Oracle this is done under the Master Scheduling module) to cancel previously recorded Work or Job Order. Optionally, a user may also include information concerning the operation in the routing at which to change the production order in the Work in Process module.

Sync Salesorder

The Sync Salesorder is an XML that facilitates the synchronization of sales or customer order information kept on separate databases throughout an enterprise. The Sync Salesorder allows the adding of new sales orders and the modification of previously established sales orders.

Add Salesorder

For organizations with automated sales force systems, customer orders may be communicated to an Order Management system using the Add Salesorder XML business document. The Add Salesorder XML communicates from one business application to one or more other business applications that a Sales Order has been added or needs to be added, depending on the business case. interface scenario. Data pertinent to an order such as customer information and address (including the Bill To and Ship To addresses), sales person information, payment terms, any miscellaneous charges, an accounting distribution, the item or product ordered along with the quantity, price and other descriptive information and the schedule of delivery may be communicated effectively using the Add Salesorder XML.

Cancel Salesorder

In order to communicate from one business application to one or more other business applications that a previous Sales Order, line, or schedule is no longer needed, the **Cancel Salesorder** XML may be used. This cancel must refer to the original document, item, and schedule. To cancel the entire order, include only the Salesorder Header information for the instance of the Salesorder to be cancelled. To cancel sales order lines and/or salesorder schedules, each line or schedule to be cancelled must be included in the occurrence of the XML with the SOLINENUM and SOSLINENUM identifiers specified respectively. If a schedule is to be cancelled, the line that the schedule refers to must be included or the schedule cannot be found. This XML may be used by a salesperson who needs to cancel a pre-recorded order in an Order Management system.

Change Salesorder

A Change Salesorder XML is a business service request that is used to request that another business application component make changes to an existing Sales Order. This change must refer to the original document and/or item ordered. The change processing assumes replacement of fields sent, with the exception of the fields for the sales order ID, the sales order line number, and the sales order schedule line number. If any of these Field Identifiers require changing, that constitutes a cancellation of the request and/or the addition of another Sales Order. This commonly causes updates to occur. A salesperson may decide to make changes to the orders already recorded in an Order Management system. This allows for flexibility in a system.

Get Salesorder and Show Salesorder

Get Salesorder is another XML business service request that enables a business application module to request information concerning a specific sales order from another business application. The reply to this XML is the Show Salesorder. There are several possible business applications in several environments that may use this capability. For example, a Sales Automation application may use this XML to ask for information from a Order Management application. Request may be made by specifying the sales order ID and may optionally request for shipment and line details.

The **Show Salesorder** XML supplies Sales Order Information to another business application module such as a sales automation system. This request may be used as a response to a Get Salesorder request or as a push notification of an event. This XML does not usually cause updates to occur. Examples of what may be returned or shown include customer information, address, sales person information, and if necessary information on each order line and schedule. A sales person may also use the data returned by a Show Salesorder XML to know the commission amount to be appropriately credited for the order, assuming that the commissions have already been calculated.

Getlist Salesorder and List Salesorder

To enable a business application module to request information containing summary information for one or more sales orders from another business application, such as Oracle Order Management, the **Getlist Salesorder** XML request may be used. The response to this request is the List Salesorder. The Getlist Salesorder also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all sales order lines for a specific item. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. This does not usually cause updates to occur. It may be used as part of a large integration scenario or as a single tool for requesting information on existing demands for goods or services. For example, a Sales Automation application may use this XML to ask for information from an order management application.

A **List Salesorder** XML is used to proactively send a listing of summary information about sales orders to one or more other applications. This does not usually cause updates to occur. It may be used as part of a large integration scenario or as a single tool for sending information concerning existing demands for goods or services. For example, an order management application may use this to respond to a request for information from a Sales Automation application.

Sync PO

Sync PO is an XML business document that facilitates keeping purchase order information synchronized on separate databases throughout an enterprise. The Sync PO XML allows the adding of new purchase orders and the modification of previously established purchase orders. There are many possible business applications in several environments that may use this capability. One example of a business integration scenario where this XML could be useful is between a purchasing system and a Material Resource Planning (MRP) system under a manufacturing scenario. The environment for this XML can be within the enterprise or outside the enterprise.

Sync Routing

Routing is the process an order must take in order to produce the finished goods. The **Sync Routing XML** is used to communicate to a business application component or system the need to create a new Routing or to update an existing Routing structure. This XML may be necessary to address the Make to Order, Assemble to Order, and Finished Goods business ordering scenarios in a Logistics to Manufacturing application integration scenario. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to communicate the requirement to synchronize the Routing necessary to build finished goods. This XML may be used individually, or as part of a larger interface scenario. Routings between a Configuration Management system and a manufacturing system or with a Finite Scheduling system are examples where the Sync Routing XML can be used. The environment for this XML can be within the enterprise or outside the enterprise.

Get Routing and Show Routing

The purpose of the **Get Routing XML** is to communicate to a business application module or system a request for an existing routing structure to be returned in a Show Routing XML. This may be used individually, or as part of a larger interface scenario.

Show Routing is an XML business service request that communicates to a business application module or system the relevant information about a specific routing. This XML may be used individually, or as part of a larger interface scenario. A Show Routing may be used to request a routing to be sent between a Routing and Configuration Management (or Oracle Bill of Materials module) system and a Manufacturing Execution System (Oracle Work in Process). This same scenario could exist between a Routing and Configuration Management system and a Finite Scheduling system. Relevant data such as the series of operations that create the routing, description of a particular item within a routing structure, a grouping of operations for the routing as well as a sequencing of operations, relationships between operations, the people needed within an operation and the description of the step within an operation for a specific routing are examples of information that may be communicated.

Getlist Routing and List Routing

A **Getlist Routing XML** is used to communicate to a business application component or module a request for a summary list of a routing structure or structures to be returned in a List Routing XML. This XML may be used individually, or as part of a larger interface scenario. Example could be a Getlist Routing to request a list of possible routings to be sent via a List Routing between a Configuration Management system and a Manufacturing Execution System. This same scenario could exist between a Configuration Management system and a Finite Scheduling system.

The purpose of the **List Routing** XML is to communicate one or more summary listings of routing information to another business application component. This may be the result of a Getlist request between a Bill of Material system and a Work in Process system or it may be initiated by some other business event.

Getlist BOM and List BOM

Getlist BOM (Bill of Material) is an XML that enables an application or component to request a summary list of Bill of Material information from another business application or component. The response to the Getlist BOM is the List BOM. The Getlist BOM also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all Bills of Material for a specific ITEM. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to communicate Item Bill of Material information.

The **List BOM** XML is used to communicate one or more summary listings of BOM information to another business application component. This may be the result of a Getlist BOM XML or it may be initiated by some other business event. This XML may return information that generally describes the Bill of Material Structure and its contents, the attributes of a specific item, the attributes of a specific OPTION for an ITEM and information that describes the class of OPTION for a particular Product or Item.

Get Item and Show Item

A **Get Item** XML enables a business application module to request information concerning a specific ITEM from another business application. The reply to this XML is the Show Item XML. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to request item information. With Oracle E-Business Suite, the Master Scheduling/MRP system uses item information to do forecasting of resources, Purchasing uses item information as basis for material replenishment and Inventory maintains these item information in its Master Item table.

To supply ITEM information to another business application module the Show Item XML can be used. This request may be used as a response to a GET ITEM request or as the result of some other business event. This XML does not usually cause updates to occur. There are many possible business applications in several environments that may use this capability. For example, item information may be communicated to a Purchasing Application from the Master Files kept in a MRP or Inventory system. The environment for this XML can be within the enterprise or outside the enterprise. Item information may include: general description of an item and its attributes, attributes of additional units-of-measure for the Item, attributes of cost or value for the Item and the location the item may be kept and the attributes of that location in relation to the item.

Getlist Item and List Item

Item is anything made, purchased, or sold including components, sub-assemblies, finished goods or supplies. A **Getlist Item** XML business document enables a business application module to request summary information concerning an item or items from another business application. For example, an MRP, Inventory, or Work in Process application could use this to request item information from a purchasing application. This type of functionality is limited to the capabilities of the responding application and needs to be determined during an implementation project. Items are defined under Oracle Inventory and item information is used by other modules such as MRP, Inventory and Work in Process. The response to this request is the List Item XML. This does not usually cause updates to occur. The environment for this XML can be within the enterprise or outside the enterprise.

The List Item XML enables a business application module to respond to a Getlist Item request or to proactively send a listing of summary information about items to one or more other applications. Attributes of an item such as the description of the packing material to be used to package the item at its stocking unit of measure, the physical characteristics of the item, the shipping material to be used to ship the item and other attributes that generally describe the item defined in Oracle Inventory, may be listed for use by another application.

Getlist Prodorder and List Prodorder

Prodorder is an XML business document that details the manufacture of a specific quantity of assembly, using specific materials and resources, in a limited time. For other applications, this may be termed as Work Order, Discrete Job or Assembly Order. With the **Getlist Prodorder** XML, a business software component such as a Receiving or Plant Data Collection system is able to request summary production order information from a Production system or Work in Process application. This XML also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all production order lines for a specific item.

The purpose of the **List Prodorder** XML request is to enable a business software component to respond to a request or to proactively send a listing of summary information about production orders to another business software component. This XML does not usually cause updates to occur. It may be used as part of a large integration scenario or as a single tool for requesting information on existing demands for goods or services. Information on individual items on the pre-determined Bill of Material structure, lot or serial number information about the final assembly defined in the production order, and the accounting information that can optionally accompany the production order may be listed.

Sync Prodorder

The purpose of the **Sync Prodorder** XML request is to notify a manufacturing application of a change in the need to make a product. Such changes can in quantity or in need by date. The business environments most likely to require this capability are an engineer to order or a configure to order manufacturing scenario. This XML can communicate a new revision or change in configuration of the product being made.

Sync Engchgordr

Engineering Change Orders (ECOs) enable control of new item revisions and bill of material changes. With one ECO, several bill of material changes can be grouped that affect one or more bills. Using Oracle Engineering, ECOs can be defined for all types of items and bills, including manufacturing and engineering items, bills and their components, planning, model, option class, and standard items, primary and alternate bills of material.

Sync Engchordr is an XML business service request that communicates to a business application module or system the need to initiate the creation of an Engineering Change Order. This XML may be necessary to address the make to order, assemble to order, or mixed mode business ordering scenarios in an order management to manufacturing application integration scenario. There are many possible business applications in several environments that may use this capability. For example, a PDM, MRP, inventory, or manufacturing business application could use this to communicate the requirement to synchronize an Engineering Change Order.

Get Engchgordr and Show Engchordr

Engineering Change Orders (ECOs) enable control of new item revisions and bill of material changes. With one ECO, several bill of material changes can be grouped that affect one or more bills. Using Oracle Engineering, ECOs can be defined for all types of items and bills, including manufacturing and engineering items, bills and their components, planning, model, option class, and standard items, primary and alternate bills of material.

Get Engchordr is an XML business service request that communicates to a business application module or system the need to produce a **Show Engchordr** XML business document for the Engineering Change Order specified in the message. This XML may be necessary to address the make to order, assemble to order, or mixed mode business ordering scenarios in an order management to manufacturing application integration scenario.

To communicate to a business application module or system the sending systems representation of a specified Engineering Change Order, the Show Engchordr XML is used. The engineering change order is an instruction from design engineering or integrated product team that the approved design has been agreed by all stakeholder departments and is the valid method of manufacture on a given date and model unit.

Confirm Engchordr

Engineering Change Orders (ECOs) enable control of new item revisions and bill of material changes. With one ECO, several bill of material changes can be grouped that affect one or more bills. Using Oracle Engineering, ECOs can be defined for all types of items and bills, including: manufacturing and engineering items, bills and their components, planning, model, option class, and standard items, primary and alternate bills of material.

The purpose of the **Confirm Engchordr** XML is to communicate to a business application module or system that the synchronization of a specified engineering change order has been completed successfully. This XML may be necessary to address the make to order, assemble to order, or mixed mode business ordering scenarios in a order management to manufacturing application integration scenario.

Sync Dspchlist

A dispatch list is a list of tasks to be done by a manufacturing execution system in order to fulfill production orders. After production orders are sent out, master scheduling creates the dispatch list. Several applications (such as finite scheduling and ERP) provide a dispatch list to a manufacturing system.

Sync Dspchlist is an XML business service request used to synchronize dispatch list (finite schedule) information. This synchronizes information about the entire WIP transaction, information concerning the specific WIP operation, or step in a routing and information concerning the resources associated with a particular WIP operation. The synchronization process can ensure that all of the components in the integration scenario have the same data necessary to communicate.

Get Dspchlist and Show Dspchlist

A dispatch list is a list of tasks to be completed by a manufacturing execution system in order to fulfill production orders. After production orders are sent out, master scheduling creates the dispatch list. Several applications (such as finite scheduling and ERP) provide a dispatch list to a manufacturing system.

The purpose of the Get Dspchlist XML business service request is to enable a business application module to request this information from another business application. The reply to this XML is SHOW. This XML does not usually cause updates to occur. This XML may be used individually, or as part of a larger integration scenario. For example, a manufacturing execution system or work in process module may get dispatch list information from a production planning module.

Show Dspchlist is an XML business document that communicates to a business application module, such as production planning, the sending systems representation of dispatch list (finite schedule) information. This XML may be used as a response to a Get Dspchlist request or as a push notification of an event. Information about the entire WIP transaction, specific WIP operation, or step in a routing and information concerning the resources associated with a particular WIP operation may be sent through the XML.

Update Dspchlist

Update Dspchlist is an XML business service request that allows a user to update or make changes to an existing dispatch list (finite schedule) information. A dispatch list is a list of tasks to be completed by a manufacturing execution system in order to fulfill production orders. After production orders are sent out, master scheduling creates the dispatch list. Several applications (such as finite scheduling and ERP) provide a dispatch list to a manufacturing system. This XML causes updates to occur.

Sync Maintorder

Sync Maintorder is an XML that ensures that all business software components in a specific integration instance have the current maintenance order information. This XML is commonly used to publish the need to create or update a Maintenance Order in a publish and subscribe integration environment. One possible scenario is the synchronization of Maintenance Order between field devices, service trucks, and so on, with a Computerized Maintenance Management System (CMMS). The synchronization process can ensure that all of the components in the integration scenario have the same data necessary to communicate. This data may include a description of the maintenance order, any accounting distribution information associated with the order, safety information related to work, location, and equipment, information on failure, cause and remedy, resources to perform the maintenance order, labor and tooling requirements to perform the operation, and information on the specific maintenance operation to be performed.

Create Maintorder

The purpose of the Create Maintorder XML document is to publish to a business application component or system the need to create or update a maintenance order. This XML is commonly used to in a publish and subscribe integration environment. This XML may be used individually, or as part of a larger interface scenario. For example, creation of a maintenance order may be sent from a field device to a Computerized Maintenance Management System (CMMS).

Update Maintorder

The purpose of the Update Maintorder XML document is to communicate any updates or changes necessary in an existing maintenance order. The environment for this XML can be within the enterprise or outside the enterprise. Maintenance orders detail the necessary maintenance operations for service equipment and tools used in the manufacturing scenario. Updates or changes may be made with the maintenance order information such as labor or craft resources, tool resources, operation, labor, and material resources or header information describing the maintenance order.

Cancel Maintorder

Cancel Maintorder is an XML business service request that communicates from business application component such as a field device to one or more other business applications (that is, a computerized maintenance management system) that a previous maintenance order is no longer needed. This cancel must refer to the original document and/or maintenance order posted.

Get Maintorder and Show Maintorder

The purpose of the Get Maintorder XML is to enable a business applications module to request order information from another business application. The response to this XML is SHOW. For example, a field device may want to get relevant information such as the resources to be used in the conduct of the maintenance operation, the operations that need to be performed under the order, and accounting distribution that may be necessary to relate the expenses of the maintenance to a particular cost center or department from a Computerized Maintenance Management System (CMMS).

In response, the CMMS creates a Show Maintorder XML to communicate data that meets the selection criteria posted under a Get Maintorder XML. The sending application sends what is requested in the Get Maintorder XML, if applicable, or alternatively, sends information that matches the data within the specified data type. A Show Maintorder XML communicates to a business application module or system the sending systems representation of maintenance order information.

Getlist Maintorder List Maintorder

A Getlist Maintorder XML enables a business application module to request information containing summary information for one or more maintenance orders. The response to this request is the LIST. The environment for this XML can be within the enterprise or outside the enterprise. The Getlist Maintorder also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all RESOURCE data type occurrences for a specific OPERATION. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. This XML does not usually cause updates to occur.

A List Maintorder XML publishes one or more summary listings of maintenance order information to other business application component. This may be in response to a Getlist Maintorder request or to proactively publish a listing of summary maintenance order information for a business event. When a receiving application receives this XML, the information can be used as is or it may be used to initiate a selection of a specific maintenance order through the Get Maintorder XML request. The processing is designed to provide multiple occurrences of summary data.

Sync UOMGroup

A UOMGROUP is a set of item-independent relationships that describe how an alternate units-of-measure is related to the stocking unit-of-measure or to other alternate units-of-measure. The environment for this XML can be within the enterprise or outside the enterprise. The purpose of the Sync UOMGROUP XML is to supply a set of unit-of-measure relationships to another business application module. This XML addresses the need for applications to exchange item-independent alternative UOM information beyond the stocking UOM. Item-independent implies that the materials management component or other subscribing applications manage the item's UOM as an entirely separate grouping that are later associated with one or more items.

This XML supports unit-of-measure (UOM) information flows associated with items being managed as inventory. The materials management, inventory control, warehousing and receiving, or shipping departments within an organization require the packaging relationship details between UOMs and the handling characteristics for each UOM to effectively manage the flow and storage of inventory. While the stocking level UOM provides for basic inventory accounting functionality, it may not be the most efficient, common, or natural UOM for manufacturing, purchasing, selling, or handling the item. The item may be packaged into standardized or supplier-specific bulk quantities for import or export, wholesale, or retail sales (for example, each, box, case, pallet, and so on).

Get UOMGroup and Show UOMGroup

A UOMGROUP is a set of item-independent relationships that describe how an alternate units-of-measure is related to the stocking unit-of-measure or to other alternate units-of-measure.

The purpose of the Get UOMGROUP XML is to request an existing UOMGROUP structure or structures from a business application component or module to be returned in a Show UOMGROUP XML. There are many possible business applications in several environments that may use this capability. For example, an MRP, inventory, or manufacturing business application could use this to request alternate UOM information for one or more items.

The Show UOMGROUP XML transfers the UOM relationships independent of the item definition. This XML supplies Unit-of-Measure Group relationship information to another business application module. This request may be issued as a response to a Get UOMGROUP request or as the result of some other business event. This XML addresses the need for applications to exchange item-independent alternative UOM information beyond the stocking UOM. Item-independent implies that the materials management component or other subscribing applications manage the item's UOM as an entirely separate grouping that are later associated with one or more items. The environment for this XML can be within or outside the enterprise.

Getlist UOMGroup and List UOMGroup

A UOMGROUP is a set of item-independent relationships that describe how an alternate units-of-measure is related to the stocking unit-of-measure or to other alternate units-of-measure.

The purpose of the Getlist UOMGROUP XML is to request a summary list of a UOMGRPHDR structure or structures from a business application component or module to be returned in a List UOMGROUP XML. The environment for this integration is application-to-application (A2A) within a single enterprise, or extended across enterprises, where there is a single master application that manages the assignment of all item identifiers. There are many possible business applications in several environments that may use this capability. For example, an MRP, inventory, or manufacturing business application could use this to request alternate UOM information for one or more items.

A List UOMGROUP XML supplies unit-of-measure group summary information to another business application module. This may be the result of a Getlist UOMGROUP request or some initiated by some other business event. When a receiving application receives this XML, the information can be used as is or it may be used to initiate a selection of a specific UOM group through the Get UOMGroup request. The processing is designed to provide multiple occurrences of summary data. The List XML is used for the receiver of the Getlist XML to respond with the results of the search that was initiated by the Getlist. Data such as those describing the packing material to be used to package the UOM, the inventory stocking unit of measure that can be tracked by an inventory control application, physical characteristics of the UOM (that is, height, volume, width), description of the shipping material to be used, are listed in summary by a List UOMGROUP XML to respond to a particular GetList XML.

Sync Catalog

Sync Catalog is an XML document that communicates the need to initiate the creation of catalog information, as well as update existing catalog information, to a business application module or system. In communicating catalog information, the Sync Catalog may cause other information to be coordinated including item identifiers, specifications, pricing information agreed (that is, purchase agreements, price lists), availability and delivery information, and related items and accessories. This XML may be necessary to address the make to order, ordering scenarios in an Order Management to Manufacturing application integration scenario.

The existing definition does not attempt to model configuration rules, and is therefore not sufficient to model configure to order environments that require cross option validation. Assemble to order environments that can be modeled using a marketing bill of material are addressed. There are many possible business applications in several environments that may use this capability. For example, a Manufacturing, distributor or reseller business application could use this to communicate the requirement to synchronize a Catalog.

It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog. A customer purchases the product from the Catalog provider. They have the capability to do comparison shopping from the catalog.

Get Catalog and Show Catalog

The purpose of the GET CATALOG XML is to enable a business application module or system to request catalog information. In communicating Catalog information, the Get Catalog XML may cause other information to be coordinated including, but not limited to Item Identifiers, Specifications, Pricing Information agreed (that is, Purchase Agreements, Price Lists), Availability and Delivery Information and related items and accessories. This XML may be necessary to address the Make to Order or ordering scenarios in an Order Management to Manufacturing application integration scenario. There are many possible business applications in several environments that may use this capability. For example, a Manufacturing, distributor, or reseller business application could use this to get Catalog information.

It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog. A customer purchases the product from the Catalog provider. They have the capability to do comparison shopping from the catalog.

A Show Catalog is an XML response to a Get Catalog request. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. The purpose of the Show Catalog XML is to supply a business application module or system with requested catalog information. Catalog information may include: fields that describe the catalog, identity of the publisher of the catalog, any category defined in the system to group specific items, description or listing of the items identified under the catalog, the specifications and specification value of a given item. The sending application sends what is requested in the Get Catalog XML, if applicable, or alternatively, sends information that matches the data in each of the data types.

Sync Itemclass

The Item Classification XML may be used in both a business to business context and an application integration context. An example of this in an application integration context might be between a Product data management system and a procurement system to access outsourcing opportunities for items in a similar classification. Another example in a business to business context may be a component supplier management company letting a supplier know the marketing classification scheme it wishes to use in its catalog.

Sync Itemclass is an XML that communicates to a business application module or system the need to synchronize the classification and specification schemes. This may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in an Order Management to a Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios.

There are many possible business applications in several environments that may use this capability. For example, a PDM, MRP, Inventory, or Manufacturing business application could use this to communicate the requirement to synchronize Item classification schemes. Examples may include, (but are not limited to): marketing classifications, production classifications, procurement classifications and shipping classifications.

Get Itemclass and Show Itemclass

The Item Classification XML may be used in both a business to business context and an application integration context. An example of this in an application integration context might be between a Product data management system and a procurement system to access outsourcing opportunities for items in a similar classification. Another example in a business to business context may be a component supplier management company letting a supplier know the marketing classification scheme it wishes to use in its catalog.

Get Itemclass is an XML business document that enables a business application module or system to request information concerning classification and specification schemes. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in an Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios.

There are many possible business applications in several environments that may use this capability. For example, a Catalog Management System could use this to communicate the requirement to get Item classification schemes information from Product Data Management, Materials Resource Planning, Inventory, or Manufacturing business applications. The CLSSSCHMID Field Identifier is used as a selection field. This is the only Field Identifier value that should be sent in this GET request.

A Show Itemclass XML is used to supply a business application module or system with information concerning classification and specification schemes. This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in a Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios. This XML returns information such as classification scheme description, data that describes an element or breakdown of the requested classification scheme, item category and item category assignments.

Sync ITEMXREF

The Item cross-reference XML may be used in both a business to business context and an application integration context. An example of this in a business to business context might be a customer letting a supplier know the item numbering for a given European Article Number. An example of this in an application integration context might be between a Product Data Management system being the source system for Harmonized Schedule B numbers that are used by the transportation management system.

Sync ITEMXREF is an XML document that communicates to a business application module or system the need to synchronize an Item cross-reference. Cross-references may be to other item identifiers to the same form fit and function, as well as references to item identifiers of other items (form fit and function). For this XML, item relationships are used to refer to where the “to item” identifier, identifies a different form, fit and function to the “from item” identifier. It should be noted that the item identifier that is “Primary” in one system may be a secondary identifier in another system.

For example, in the Application Integration space, the manufacturing system may regard the “Item Number” as the primary identifier. The Order Management System may regard the Catalog number as the primary identifier. A company that manufactures hand held multi-meters may identify a given item in manufacturing with a 12 digit numeric code, 5432 123 12345. The marketing and sales department may refer to the same item by its catalog number of FL 30/4.

In the Business to Business case a supplier of hand held multi-meters may market their products through a catalog provider. The supplier has an item identifier with a corresponding party specific cross reference to the catalog providers identifier for the same item. The catalog provider has a item identifier for hand held multi-meters and a corresponding party specific cross reference to the suppliers item number. An example of this in a business to business context might be a customer letting a supplier know the valid substitutes that a supplier may supply to fulfill an order for a specific item number. An example of this in an application integration context might be between a Product Data Management (PDM) system and an Order Management system for accessories that may be offered to a customer with the sales of a major item. For example, if a designer of a video camera has designed it to work with accessories such as tripod, extended life battery pack, external microphone and head cleaner, the video may be designed to have the spares replaced by the consumer such as lens cover, strap and handle. The video camera may need the following consumable items on a recurring basis: video cassettes, batteries or lens cleaners. The relationship between these items and the video camera may need to be represented to the Customer in Web Based ordering or Customer Service Representative (CSR), in desk based order entry.

Get ITEMXREF and Show ITEMXREF

The purpose of the Get ITEMXREF XML is to enable a business application module or system to request information concerning an Item cross-reference. Cross-references may be to other item identifiers to the same form fit and function, as well as references to item identifiers of other items (form fit and function). For this XML, item relationships are used to refer to where the “to item” identifier, identifies a different form, fit and function to the “from item” identifier. It should be noted that the item identifier that is “Primary” in one system may be a secondary identifier in another system. For example, in the Application Integration space, the manufacturing system may regard the “item Number” as the primary identifier. The Order Management System may regard the Catalog number as the primary identifier. A company that manufacture hand held multi-meters may identify a given item in manufacturing with a 12 digit numeric code, 5432 123 12345. The marketing and sales department may refer to the same item by its catalog number of FL 30/4.

This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in a Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios. The Catalog Management System quotes a set of cross references that are not currently in the catalog. The cross-references may be either alternate identifiers of the products in the Catalog such as EAN Numbers and UPC Codes or they may be related items such as accessories, spares, or consumables. The Cross Reference document may flow from either the purchasing system of the selling system to let the Catalog Management System be aware of the identifiers used by all buying and selling parties.

A Show ITEMXREF XML supplies a business application module or system with information concerning an Item cross-reference. The XML may be used to relate item identifiers for item identifiers that identify different items (form fit and function). The Relationship types may also be universal. The sending application sends what is requested in the GET, if applicable, or alternatively, sends information that matches the data in the specified Data Type. The Show ITEMXREF XML allows the user to cross-refer either between classifications, or between items.

Sync Pricelist

A Price List is a list containing the basic selling price per unit for a group of items, item categories, or services offered. Price lists are essential to ordering products because each item entered on an order must have a price. Most Order Management systems contain basic list information and one or more pricing lines, pricing attributes, qualifiers, and secondary price lists. Basic information for Oracle Order Management includes the price list name, effective dates, currency, pricing controls, rounding factor, and shipping defaults such as freight terms and freight carrier.

Sync Pricelist is an XML document that communicates to a business application module or system the need to synchronize product price list information. This allows the publication from the order management system of the Price List that accompanies the catalog. This XML may be necessary to address the Make to Order, Assemble to Order, or Mixed Mode business ordering scenarios in an Order Management to a Manufacturing application integration scenario.

There are many possible business applications in several environments that may use this capability. For example, a Manufacturing, distributor or reseller business application could use this to communicate the price change or request a price list. It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog. A customer purchases the product from the Catalog provider. They have the capability to do comparison shopping from the catalog.

The primary business process the Pricelist XML supports are to provide an instruction from a supplier to a customer that the supplier's list price for the suppliers product needs updating. It may also be an instruction from Marketing Systems to update order management systems. It may be an instruction from Order Management systems to Sales force automation systems to update Price Lists.

Get Pricelist and Show Pricelist

A Price List is a list containing the basic selling price per unit for a group of items, item categories, or services offered. Price lists are essential to ordering products because each item entered on an order must have a price. Most Order Management systems contain basic list information and one or more pricing lines, pricing attributes, qualifiers, and secondary price lists. Basic information for Oracle Order Management includes the price list name, effective dates, currency, pricing controls, rounding factor, and shipping defaults such as freight terms and freight carrier.

The Get Pricelist XML business document enables a business application module or system to request information concerning new or existing product price lists. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. This XML may be necessary to address the Make to Order, Assemble to Order, or Mixed Mode business ordering scenarios in an Order Management to Manufacturing application integration scenario.

There are many possible business applications in several environments that may use this capability. For example, a Manufacturing, distributor or reseller business application could use this to request a price list. The Catalog Management System quotes a price list that does not exist in the Catalog Management System causing the Catalog Management System to request the price list. It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog. A customer purchases the product from the Catalog provider. They have the capability to do comparison shopping from the catalog. Selection of price list to be returned may be made by providing a Pricelist ID using the Get Pricelist XML.

The purpose of the Show Pricelist XML is to supply a business application module or system with information concerning new or existing product price lists. This allows for the publication from the order management system of the Price List that accompanies the catalog.

The following types of data may be returned in a Show Pricelist XML:

- Price List Lines—the list of items or commodities and a base price for the items
- List price breaks—the prices and modifiers to the price for buying a given quantity or value of an item or item category on a price list line
- List Price Breaks—the prices and modifiers to the price for buying a given value of any product
- Price List Qualifiers—qualifies the Price Lists that may be used to price an item or item classification, in a given catalog, on a given date or for a given customer.

Sync Itemspecs

The Item Specification XML may be used in both a business to business context and an application integration context. An example of this in an application integration context might be between a Product Data Management (PDM) system and a Procurement system to access outsourcing opportunities for items with similar specifications.

Sync ITEMSPECS is an XML document that communicates to a business application module or system the need to synchronize the specification of items within a catalog. The Item specification describes items in a catalog. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners.

This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in a Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios. There are many possible business applications in several environments that may use this capability. For example, a Product Data Management, Material Resource Planning, Inventory, or Manufacturing business applications could use this to communicate the requirement to synchronize Item specifications. Examples may include, (but are not limited to): Engineering systems communicating specification information to a purchasing system or a Supplier to a CSM company letting the CSM know the specifications of the items in the supplier's catalog.

Get Itemspecs and Show Itemspecs

The purpose of the Get ITEMSPECS XML is to enable a business application module or system to request information concerning the specification of items within a catalog. The Item specification describes items in a catalog. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in a Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios. An example scenario for the Get ITEMSPECS XML could be that the specification documents quoted in the catalog are not yet in the Catalog Management System causing the Catalog Management System to request the item specifications from the Product Data Management system (PDM).

Using a Show ITEMSPECS XML, the PDM system publishes a set of Specifications for the items in the catalog. The purpose of the Show ITEMSPECS XML is to supply a business application module or system with information concerning the specification of items within a catalog. The Item specification describes items in a catalog. This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in a Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios.

The sending application sends what is requested in the GET, if applicable, or alternatively, sends information that matches the data in the following data types:

- Feature Value information - defines the valid list of values for a given feature. An example of a feature value for a feature might be, operating voltage 110/240V AC.
- Feature Value Assignment information - describes the assignment of specifications or the values of specifications to a given item, or item category. An example of Feature Value Assignment information might be that a FL 856 hand held multi-meter may have an operating voltage of 110/240V AC.

Sync RFQ

Request for Quotation (RFQ) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Sync RFQ XML refers to is the integration of a buyer's business software components to supplier's business software components.

Sync RFQ XML document ensures that all business software components in a specific integration instance have the current Request for Quotation information. Sync RFQ XML may be used in an integration scenario between a Buyer's Purchasing system and a Seller's Order Management system. The workflow sequence for custom, built to order involves more activity than that of finished goods. The buyer's focus involves finding the best price for an item or quantity of items that are not currently available. A long-term business relationship is often involved leading to expectations that must be addressed within the quote. Customized products require a higher level of specification. Individual supplier inquiries may be answered formally for all participants to insure competitive parity. Conversely, buyers may have questions related to an individual quote that lead to a change. Sync RFQ XML is used when the RFQ data persists in multiple systems.

Add RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Add RFQ XML refers to is the integration of a buyer's business software components to supplier's business software components.

Add RFQ is an XML used to communicate from one business application to one or more other business applications that additional data related to a Request for Quotation has been added or needs to be added, depending on the business case. The environment for this XML can be within the enterprise or outside the enterprise. A Buyer's Purchasing System may use the Add RFQ XML to add a Request for Quotation to a Supplier's Order Management system.

The Add RFQ XML is used in pre-purchase activities for Quantities of Finished Items and Custom Build to Order Items. Add RFQ XML releases a request to one or more partners providing deadlines for response and item details. Add RFQ XML is used when the recipient takes ownership of the RFQ source data.

Change RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Change RFQ XML refers to is the integration of a buyer's business software components to supplier's business software components.

A Change RFQ XML requests that another business application component make changes to an existing Request for Quotation. The environment for this XML can be within the enterprise or outside the enterprise. A request is modified to include answers to questions or clarification of specifications using a Change RFQ XML.

Cancel RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Cancel RFQ XML refers to is the integration of a buyer's business software components to supplier's business software components.

The purpose of the Cancel RFQ XML is to publish to a business application or system the need to cancel an entire Request for Quotation or one or more of its line items. The environment for this XML can be within the enterprise or outside the enterprise. A Buyer organization's Purchasing system may send a Cancel RFQ XML to a Supplier organization's Order Management system to cancel an existing RFQ or lines of a RFQ.

Respond RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Respond RFQ XML refers to is the integration of a buyer's business software components to supplier's business software components.

A Respond RFQ XML communicates from one business application to one or more other business applications that additional data related to a Request for Quotation is available. The environment for this XML can be within the enterprise or outside the enterprise. The Respond RFQ XML is typically involved in a Custom Build to Order workflow. The workflow sequence for custom, built to order involves more activity than that of finished goods. The buyer's focus involves finding the best price for an item or quantity of items that are not currently available. A long-term business relationship is often involved leading to expectations that must be addressed within the quote. Customized products require a higher level of specification. Individual supplier inquiries may be answered formally for all participants to insure competitive parity. Conversely, buyers may have questions related to an individual quotation that lead to a change. Suppliers respond with questions and/or clarification issues using a Respond RFQ XML.

Getlist RFQ and List RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Getlist RFQ XML refers to is the integration of a buyer's business software components to supplier's business software components.

Getlist RFQ is an XML that enables a business application module to request information containing summary information for one or more Request for Quotations. The response to this request is LIST RFQ. The environment for this XML can be within the enterprise or outside the enterprise. The Getlist RFQ also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all QUALFICATN Data Type occurrences for a specific RFQLINE. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. The typical scenario for use of this XML is that a Supplier or Buyer requests a list of RFQs.

The List is sent to the requestor by using a List RFQ XML. This XML publishes one or more summary listings of Request for Quotation information to other business application component. This may be in response to a Getlist RFQ request or to proactively publish a listing of summary Request for Quotation information for a business event. When a receiving application receives this XML, the information can be used as is or it may be used to initiate a selection of a specific Request for Quotation through the Get RFQ XML. The processing is designed to provide multiple occurrences of summary data.

Get RFQ and Show RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory or Work in Process systems. Another scenario of which the Get RFQ and Show RFQ XML refer to is the integration of a buyer's business software components to supplier's business software components.

Get RFQ XML document enables a business applications module to request Request for Quotation (RFQ) information from another business application. The response to this XML is a Show RFQ. The environment for this XML can be within the enterprise or outside the enterprise. A Supplier's Order Management system sends a Get RFQ XML to a Buyer's Purchasing system to request RFQ information. It may also be that intermediaries are used to manage RFQ exchange activities between a Buyer and a Seller. The workflow sequence for scenarios using intermediaries involves additional activity. An intermediary may be independent or an extension of a large organization. Much of the additional activity follows from the aggregation provided by an intermediary. Multiple RFQs could reside with the intermediary. The Get RFQ XML may be used to select one of the items returned by a List RFQ XML for review. The full RFQ is sent by using a Show RFQ XML.

The Show RFQ XML communicates to a business application module or system the sending systems representation of Request for Quotation information. This request may be used as a response to a Get RFQ XML or as a push notification of an event.

Sync Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. The integration scenario for the Sync Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

A Sync Quote XML ensures that all business software components in a specific integration instance have the current Quotation information. This XML is commonly used to publish the need to create or update a Quotation in a publish and subscribe integration environment. The scenario of which the Sync Quote XML refers to is the integration of a Buyer's business software components to Supplier's business software components.

Add Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. The integration scenario for the Add Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

An Add Quote XML indicates a response to a supplier provided quotation or buyer question on a quotation. The purpose of the Add Quote XML is to communicate from one business application to one or more other business applications that additional data related to a quote has been added or needs to be added, depending on the business case. This XML commonly causes updates to occur. This XML may be used individually, or as part of a larger integration scenario. An example is the sending of an Add Quote XML from a Supplier's Order Management system to a Buyer's Purchasing system, or through an intermediary.

Change Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. A quotation often represents a binding document that may change based on negotiations between the buyer or RFQ "owner" and the seller responding to the RFQ. The integration scenario for the Change Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

The purpose of the Change Quote XML is to request that another business application component make changes to an existing Quotation. The environment for this XML can be within the enterprise or outside the enterprise. The Change Quote XML is sent by a Supplier's Order Management system to a Buyer's Purchasing system, or through an intermediary.

The Change Quote XML is used in pre-purchase activities for Quantities of Finished Items and Custom Build to Order Items. Using a Change Quote XML a Supplier may alter a quotation during negotiations in an effort to win the order.

Cancel Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. A quotation often represents a binding document that may change based on negotiations between the buyer or RFQ "owner" and the seller responding to the RFQ. The integration scenario for the Cancel Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

Cancel Quote XML is used to publish to a business application or system the need to cancel an entire quotation or one or more of its line items. The environment for this XML can be within the enterprise or outside the enterprise. This XML commonly causes updates to occur.

Respond Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. The integration scenario for the Respond Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

Respond Quote XML indicates a response to a supplier-provided quotation or buyer question on a quotation. The purpose of the Respond Quote XML is to communicate from one business application to one or more other business applications that an additional data related to a quotation has been added or needs to be added, depending on the business case. This XML commonly causes updates to occur.

Getlist Quote and List Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. The integration scenario for the Getlist and List Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

Getlist Quote XML enables a business application module to request information containing summary information for one or more quotations. The response to this request is LIST QUOTE. The environment for this XML can be within the enterprise or outside the enterprise. The Getlist Quote XML also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all SALESINFO Data Type occurrences for a specific QTELINE. This type of functionality is limited to the capabilities of the responding application and needs to be determined during an implementation project. This XML does not usually cause updates to occur. This XML may be used individually, or as part of a larger integration scenario. There are many possible business applications in several environments that may use this capability. An example is a Buyer's Purchasing system sending a Getlist Quote XML to a Supplier's Order Management system, or through an intermediary.

The purpose of the List Quote XML is to publish one or more summary listings of quotation information to other business application component. This may be in response to a Getlist Quote request or to proactively publish a listing of summary quotation information for a business event. The environment for this XML can be within the enterprise or outside the enterprise. When a receiving application receives this XML, the information can be used as is or it may be used to initiate a selection of a specific Quote through the GET QUOTE request. The processing is designed to provide multiple occurrences of summary data.

Get Quote and Show Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. The integration scenario for the Get and Show Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

Get Quote XML enables a business applications module to request this quotation information from another business application. The response to this XML is the SHOW QUOTE. The environment for this XML: can be within the enterprise or outside the enterprise. This XML does not usually cause updates to occur.

This XML may be used individually, or as part of a larger integration scenario. There are many possible business applications in several environments that may use this capability. An example is a Buyer's Purchasing system sending a Get Quote XML to a Supplier's Order Management system, or through an intermediary.

The purpose of the Show Quote XML is to communicate to a business application module or system the sending systems representation of quotation information. This request may be used as a response to a GET QUOTE request or as a push notification of an event. The environment for this XML can be within the enterprise or outside the enterprise.

Show Shipment

Shipment is an XML business document that details the intent to transport a specific quantity of material goods from a supplier to a single customer business partner destination. The Shipment has been modeled after similar proprietary documents on popular business software packages (the Oracle E-Business Suite Delivery document, the SAP Delivery Note, and so on). The environment for this XML can be within the enterprise or outside the enterprise.

A Shipment is typically derived from the shipping schedule associated with a customer's purchase or sales order, once overall demand and various other business factors which prioritize the availability of the supplier's goods inventory have been evaluated. The Shipment document is designed to have a dynamic structure & content. Initial shipment planning information can be updated and significant detail (actual picked inventory attributes, ship unit packaging, and so on) may be added during the execution phase of the supplier's order fulfillment and shipping business processes. The final form of the Shipment document provides detail about the carrier and level of service used to transport the material, the exact quantity and attributes of the material shipped, and how that material is physically packaged and identified for transport.

To aid the customer's planning and receiving business processes, the supplier may transmit the final Shipment document to customer in advance so that they can prepare for carrier arrival and then efficiently accept and utilize the ordered material. In this use case, the Shipment document may function as a traditional Advance Ship Notice (ASN).

This XML may be used individually, or as part of a larger interface scenario. For example, a Shipping Management System uses a Show Shipment XML to show shipment information to a Buyer's Purchasing System where the receipt of goods are recorded.

Sync Planschd

PLANSCHD indicates a demand forecast sent from a customer to a supplier, or a supply schedule sent from a supplier to a customer. Sync Planschd XML communicates requirement information (part number, quantity, and so on) between a customer and their supplier on a regular basis, for example daily, weekly and so on, or for a user-defined time bucket type definition that is sent as part of this PLANSCHD.

Since collaboration between a customer and a supplier can potentially go through several rounds of negotiations, both parties can use the same XML to communicate their current demand or supply schedule in response to what they received from the other party. Either party can indicate detailed exception descriptions along with reason code and action code to the other party why the original demand or supply requirements need to be adjusted.

The Sync Planschd XML may be used individually, or as part of a larger interface scenario. An example would be the synchronization of plan schedules between a Buyer's Scheduling application and the Supplier's Release Management application. The Supplier Scheduling Application maintains schedules regarding planned demand, shipping and sequence of delivery. The Release management Application drives the MRP, the shipment and the sequence schedules in the Suppliers ERP application.

Get Planschd and Show Planschd

PLANSCHD indicates a demand forecast sent from a customer to a supplier, or a supply schedule sent from a supplier to a customer. The Get Planschd XML enables business applications module to request plan schedule information from another business application. The response to this XML is Show PLANSCHD.

Since collaboration between a customer and a supplier can potentially go through several rounds of negotiations, both parties can use the same XML to communicate their current demand or supply schedule in response to what they received from the other party. Either party can indicate detailed exception descriptions along with reason code and action code to the other party why the original demand or supply requirements need to be adjusted.

This XML may be used individually, or as part of a larger interface scenario. An example would be the Supplier's Release Management application sending a Get Planschd XML to a Buyer's Scheduling application. It could be that the supplier Management application quotes a planning schedule that does not exist in the Demand Management application causing the Demand Management Application to request the planning schedule. This interface scenario assumes that the Supplier Scheduling Application maintains schedules regarding planned demand, shipping and sequence of delivery. The Release management Application drives the MRP, the shipment and the sequence schedules in the Suppliers ERP application.

Show Planschd XML communicates to a business application module or system the sending systems representation of plan schedule information. This request may be used as a response to a Get Planschd XML or as a push notification of an event. This XML is used to publish planning schedule from a Supplier Scheduling application.

Sync Seqschd (Sequence Schedule)

Sync Seqschd XML enables the exchange of sequence schedule information authorizing a sequenced shipment of parts for specific trading partners and addresses. The environment for this XML can be within the enterprise or outside the enterprise. Commonly, the sequence schedule is generated by a work in process application and transmitted to an order or material planning application. This XML commonly causes updates to occur.

The Sync Seqschd XML may be used individually, or as part of a larger interface scenario. This event is the publication from the supplier scheduling application of a Shipment schedule to the buyer release management application. This interface scenario assumes that the Supplier Scheduling Application maintains schedules regarding planned demand, shipping and sequence of delivery. The Release management Application drives the MRP, the shipment and the sequence schedules in the Suppliers ERP application.

Get Seqschd (Sequence Schedule) and Show Seqschd (Sequence Schedule)

A Get Seqschd XML enables a Release Management application to request sequence schedule information from another business application such as a Supplier Scheduling application. The response to this XML is Show SEQSCHD. This XML does not usually cause updates to occur.

Commonly, the sequence schedule is generated by a work in process application and transmitted to an order or material planning application. The environment for this XML can be within the enterprise or outside the enterprise. The SCHEDULEID Field Identifier is used as a selection field. This is the only Field Identifier value that is used to send a Get Seqschd XML.

A Show Seqschr XML communicates to a business application module or system the sending systems representation of sequence schedule information. This request may be used as a response to a Get Seqschr request or as a push notification of an event. Information detailing general description of partners, including addresses and contacts, sequence schedule date and time information, list of items ordered along with the quantity, delivery date and other schedule information is data that may be returned with a Show Seqschr XML. The information obtained may be used by a Demand Management application to do demand forecasting.

Sync Shpschr (Shipment Schedule)

Sync Shpschr XML enables the exchange of shipment schedule information, authorizing a shipment quantity and date for specific trading partners and addresses. Commonly, the ship schedule is generated by a material planning application and transmitted to an order management application. Shipment schedules associated with a customer's purchase or sales order is used to typically derive a shipment. This is completed once overall demand and various other business factors which prioritize the availability of the supplier's goods inventory have been evaluated.

The Sync Shpschr XML is used in an interface scenario between a Supplier Scheduling Application and a Release Management application. This interface scenario assumes that the Supplier Scheduling Application maintains schedules regarding planned demand, shipping and sequence of delivery. The Release management Application drives the MRP, the shipment and the sequence schedules in the Suppliers ERP application.

Get Shpschr (Shipment Schedule) and Show Shpschr (Shipment Schedule)

The Get Shpschr XML enables a Seller's Release Management application to request shipment schedule information from a Buyer's Supplier Scheduling application. This interface scenario assumes that the Supplier Scheduling Application maintains schedules regarding planned demand, shipping and sequence of delivery. The Release management Application drives the MRP, the shipment and the sequence schedules in the Suppliers ERP application. The response to this XML is the SHOW SHPSCHR.

The Show Shpschr XML communicates to a business application module or system the sending systems representation of SHPSCHR information. This request may be used as a response to a GET SHPSCHR request or as a push notification of an event. Information on the partner including the location and contacts, the items ordered along with the quantity, delivery date and other schedule information and line item exceptions based on predefined business rules or contract agreements may be returned using a Show Shpschr XML.

Process Invoice

Process Invoice is an XML that transmits an invoice from a supplier to a customer, thus involving an interface between a Supplier's Accounts Receivable application with a Customer's Accounts Payable application. Invoice in this interface scenario refers to either an item based customer invoice or as a general purpose customer credit or debit memo. A customer invoice is a document that is created in Receivables that lists amounts owed for the purchases of goods or services. This document also lists any tax, freight charges and payment terms. The Process Invoice XML transmits invoice information such as line items, charges, allowances and tax, partner information including location and contacts, invoice payment terms and information about a business partner document reference (that is, customer's purchase order, supplier's sales order) for the invoice as a whole or for a particular invoice line.

Process WIPSLIT

Process WIPSLIT (Work in Process Split) is an XML business service request that notifies a Manufacturing Application of the creation of multiple production lots from a single production lot of a product being made on a production order. The business environment most likely to require this capability is a lot-based discrete manufacturing scenario. The environment for this XML is normally within the enterprise. Process WIPSLIT XML may be used in an integration that involves a manufacturing execution system (MES) and a central ERP system. The MES is the source of production events and activities and the ERP (with Order Management, Bill of Material, Cost Management and Inventory) is the source of related setup events which occur prior to production.

The Process WIPSLIT XML communicates the originating lot, the resulting lots, their quantities, and the processing step at which this event occurred, along with the time at which this event occurred. This XML commonly causes updates to occur. Process WIPSLIT XML may be used individually, or as part of a larger interface scenario. The workflow could start with the placement and synchronization of orders in the ERP system and the issuance of materials and charging of resources to the production order. The production order moves through the operations on the shop floor and throughout the production process, the order may experience a number of moves and splits or merges, and the creation of one or more bonus lots of material may possibly occur. The result of a WIP split may cause changes in the cost of the finished goods.

Process WIPMERGE

Process WIPMERGE (Work in Process Merge) is an XML business service request that notifies a Manufacturing Application of the creation of a single production lot from multiple production lots of a product being made on a production order. The business environment most likely to require this capability is a lot-based discrete manufacturing scenario. The environment for this XML is normally within the enterprise. Process WIPMERGE XML may be used in an integration that involves a manufacturing execution system (MES) and a central ERP system. The MES is the source of production events and activities and the ERP (with Order Management, Bill of Material, Cost Management and Inventory) is the source of related setup events which occur prior to production.

The Process WIPMERGE XML communicates the originating lots, the resulting lot, lot quantities, and the processing step at which this event occurred, along with the time at which this event occurred. This XML commonly causes updates to occur. This XML may be used individually, or as part of a larger interface scenario. The workflow could start with the placement and synchronization of orders in the ERP system and the issuance of materials and charging of resources to the production order. The production order moves through the operations on the shop floor and throughout the production process, the order may experience a number of moves and splits or merges, and the creation of one or more bonus lots of material may possibly occur. The result of a WIP merge may cause changes in the cost of the finished goods.

Process WIPRECOVER

Process WIPRECOVER is an XML business service request that notifies a Manufacturing Application of the creation of usable production materials from material previously considered unsuitable for production use. This is most often likely to represent a return to production of scrap material. The business environments most likely to require this capability are any type of manufacturing scenario (that is, discrete, process).

This XML communicates what is being recovered, the quantity being recovered, and the processing step at which the recovered material is to re-enter the production process, along with the time at which this event occurred. This XML commonly causes updates to occur. This XML may be used individually, or as part of a larger interface scenario. Process WIPRECOVER XML may be used in an integration that involves a manufacturing execution system (MES) and a central ERP system. The MES is the source of production events and activities and the ERP (with Order Management, Bill of Material, Cost Management and Inventory) is the source of related setup events which occur prior to production.

Get WIPSTATUS and Show WIPSTATUS

Get WIPSTATUS is an XML business service request that enables a business application module to request information concerning the progress of a production order at a point in time from another business application. The business environments most likely to require this capability are any type of manufacturing scenario (that is, process, discrete) where business service requests for individual manufacturing transactions and events are not utilized. In a discrete manufacturing scenario, discrete job statuses describe the various stages in the life cycle of a job and control the activities that one can perform on it. Examples of Work in Process statuses are Complete, Released, Unreleased, On Hold, Cancelled or Close. The environment for this XML can be within the enterprise or outside the enterprise.

The Get WIPSTATUS XML communicates what quantities of end product reside at which processing steps along with the time this snapshot view was taken. The response to this XML is Show WIPSTATUS. This XML does not usually cause updates to occur. This XML may be used individually, or as part of a larger interface scenario. For example, a Work in Process system sends a Get WIPSTATUS XML to a Manufacturing Execution System to request the status or progress of a production order.

To respond to a Get WIPSTATUS XML, the Show WIPSTATUS XML is used. This Show WIPSTATUS XML notifies a Manufacturing Application of the progress of a production order at a point in time. The business environments most likely to require this capability are any type of manufacturing scenario (discrete, process) where business service requests for individual manufacturing transactions and events are not utilized. The environment for this XML can be within the enterprise or outside the enterprise. This XML communicates what quantities of end product reside at which processing steps along with the time this snapshot view was taken. This XML commonly causes updates to occur.

Process WIPMOVE

Process WIPMOVE (Work in Process Move) is an XML business service request that notifies a Manufacturing Application of the progression through the production processing steps or operations of a product being made on a production order. The business environments most likely to require this capability are any type of manufacturing scenario (that is, discrete, process). The environment for this XML is normally within the enterprise. Process WIPMOVE XML may be used in an integration that involves a manufacturing execution system (MES) and a central ERP system. The MES is the source of production events and activities and the ERP (with Order Management, Bill of Material, Cost Management and Inventory) is the source of related setup events which occur prior to production.

The Process WIPMOVE XML communicates which processing step the product is coming from and which step it is being moved to, along with the quantity moving and the time this event occurred. This XML assumes that the applications involved in this business scenario have already synchronized the production item and its BOM or Routing information. This XML commonly causes updates to occur.

Process WIPMOVE XML may be used individually, or as part of a larger interface scenario. The workflow could start with the placement and synchronization of orders in the ERP system and the issuance of materials and charging of resources to the production order. The production order moves through the operations on the shop floor and throughout the production process, the order may experience a number of moves and splits or merges, and the creation of one or more bonus lots of material may possibly occur. In an environment such as Oracle Manufacturing, move transactions moves assemblies within an operation, such as from Queue to Run, or from operation to the next. Move transactions can charge resources and overheads.

Allocate Resource

Allocate Resource is an XML that notifies a Manufacturing Application of the use of required labor or machine time on a production order making a product. The business environments most likely to require this capability are any type of manufacturing scenario (that is, discrete, process).

The Allocate Resource XML communicates what machine was utilized or which person performed the work and their labor skill class, along with the amount of time worked and at what time this event occurred. This XML commonly causes updates to occur. This XML may be used individually, or as part of a larger interface scenario. For example, interface between a Manufacturing Execution System and a Cost Management system to make updates to the cost of a product with the allocated resources. Information as to the resource transaction, work order the resource is to be charged against, resources to be charged, the production order against which the resources are to be charged, the lot and/or serial number information on the production items, the operation at which resources are to be charged, and the resources (worker, machine and tools) required to perform the operation assists in allocating the right resources for the order.

Get Customer and Show Customer

Get Customer is an Way XML used to request customer information. This is likely between an Order Management system getting customer information from a customer management system. In Oracle E-Business Suite, customer records are maintained and managed in the Oracle Accounts Receivable system. Customer records include information on the customer (that is, customer name, payment method, partner type), customer's addresses and contacts. This XML does not cause updates to occur. An order in an Order Entry system can only be processed when customer records are in place and can be validated.

As a response to a GET Customer XML or a trigger by a business event, the Show Customer XML is used. The purpose of the Show Customer XML is to provide Customer information to a requesting business application, such as an Order Management system. This XML does not usually cause updates to occur.

Get Supplier and Show Supplier

Get Supplier is an Way XML used to request supplier information. This is likely between an Order Management system getting supplier information from a supplier management system. In Oracle E-Business Suite, supplier records are maintained and managed in the Oracle Purchasing system. Supplier records include information on the supplier (that is, supplier ID, currency, payment method, partner type), supplier's addresses and contacts. This XML does not cause updates to occur.

The purpose of the Show Supplier XML is to provide Supplier information to a requesting business application, such as Order Management system. This XML may be in response to a Get Supplier XML, or it may be triggered by a business event. This XML does not usually cause updates to occur.

Sync Ecatalog

Sync ECATALOG is an XML that synchronizes catalog information between two systems. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners.

There are many possible business applications in several environments that may use this capability. Some examples of usage scenarios are a Manufacturer synchronizing catalogs with distributors, suppliers, or e-marketplaces and distributors, suppliers, or e-marketplaces synchronizing catalogs with buyers or other trading partners. It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog.

The existing definition does not attempt to model configuration rules, and is therefore not sufficient to model configure to order environments that require cross option validation. Assemble to order environments that can be modeled using a marketing bill of material is addressed. The Catalog exchange scenario can be implemented either as a simple scenario using a single XML, or in the case of large catalogs involving complex pricing scenarios or partner specific details, as multiple XMLs.

Get ECATALOG and Show ECATALOG

Get ECATALOG, is an XML that enables a business application module or system to request catalog information. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. The Catalog information that is requested by the Get CATALOG XML may include item identifiers, specifications, pricing information agreed on, price lists and availability and delivery information.

There are many possible business applications in several environments that may use this capability. Some examples of usage scenarios are Manufacturer exchanging catalogs with distributors, suppliers, or e-marketplaces and distributors, suppliers or e-marketplaces exchanging catalogs with buyers or other trading partners. It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog. The Catalog exchange scenario can be implemented either as a simple scenario using a single XML, or in the case of large catalogs involving complex pricing scenarios or partner specific details, as multiple XMLs.

The purpose of the Show ECATALOG XML is to supply a business application module such as a Catalog Management System with requested catalog information. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. A Catalog Publisher should be able to publish catalog information to a customer or agent on information such as description of the product or service, party specific part identifiers, pricing information agreed on and other information depending on the needs of the subscriber. A Catalog Searcher should be able to request catalog information for a part or service identifier including description of product or service, part specific part numbers and other relevant information to the catalog.

Sync Invoice

Sync Invoice is an XML that synchronizes or transmits an invoice from a supplier to a customer, thus involving an interface between a Supplier's Billing system with a Customer's Accounts Payable application. Invoice in this interface scenario refers to either an item-based customer invoice or as a general purpose customer credit or debit memo. A customer invoice is a document that is created in Receivables that lists amounts owed for the purchases of goods or services. This document also lists any tax, freight charges and payment terms. The Sync Invoice XML synchronizes invoice information such as line items, charges, allowances and tax, partner information including location and contacts, invoice payment terms and information about a business partner document reference (that is, customer's purchase order, supplier's sales order) for the invoice as a whole or for a particular invoice line.

Get Invoice and Show Invoice

A Get Invoice XML enables a request of a customer invoice. This XML may be used as a request by a Customer to its Supplier. The interface would involve a Customer's Accounts Payable system requesting for invoice information from a Supplier's billing application. This allows the customer to get details of the invoice for processing and payment. A billing invoice may include information such as line items, charges, allowances and tax, partner information including location and contacts, invoice payment terms and information about a business partner document reference.

A response from the Supplier's billing system is to be carried through a Show Invoice XML. A Show Invoice XML transmits an invoice from a supplier to a customer. This XML may be used as a response to a Get Invoice request or as a push notification of an event.

Get Consumption and Show Consumption

The most common use of the Get Consumptn XML is to request a buyer's usage information about an item or product for the supplier of such item or product. This XML does not create or update either buyer's or supplier's inventory records. The receiver of the request is responsible to make effective use of this information. The Get Consumption XML may be used for a supplier of goods to request from the buyer the consumption status of goods. This XML may also be used for a vendor to request from the retailer if retail sales of goods have been made and for inventory systems to request consumption status from plant data collection and warehouse management systems.

The most common use of the Show Consumtln XML is to share a buyer's usage information about an item or product with the supplier of such item or product. This XML does not create or update either buyer's or supplier's inventory records. The receiver of the request is responsible to make effective use of this information.

This is an outline of the business flow that these XML supports:

1. Overall purchase, replenishment or vendor managed inventory agreement is in place and/or a Get Consumptn message is sent by the supplier.
2. Show Consumptn Message is returned to the supplier, distributor or third party logistics provider, that material has been consumed. This is done in response to events such as these (and/or the Get message), depending on implementation context:
3. Material is replenished to line side at manufacturing facility.
4. Material is assembled into final product.
5. Material is purchased and removed from facility by customer.

6. Supplier, distributor, third party logistics provider replenishes material, using information provided in the Show Consump^{tn} message, the demand and shipment forecasts, and the terms of the overall purchase or vendor managed inventory agreement.

Create Requisitn

Create Requisitn is an XML request that notifies another business application of the need to order parts in a specific quantity, for a specific need by date. This XML usually causes updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand. A Material Resource Planning (MRP) or an Inventory system may send a Create Requistn XML to a Purchasing System that handles purchase requisitions and orders maybe because reorder points indicate a need to request materials for production. Requisitions signify demand for goods and services to another business application for consideration of buying or in some way obtaining requested items. In Oracle, this facilitates the sending of a purchase requisition to trigger the issuance of a purchase order for items requested.

Getlist LDGRACTUAL and List LDGRACTUAL

A **Getlist LDGRACTUAL** is an XML that requests information containing summary information for one or more ledgers. Many applications in the enterprise environment create data that causes changes in the account balances of a general ledger application (that is, Benefits, Costing, Human Resources and Payroll, Inventory, Manufacturing, Production, Treasury). The integration scenario for this XML is for enterprise applications business software components to obtain and receive accounting data from the general ledger business software component. The request or response style of messaging which the enterprise application sends a request message and expects to receive a response message back. The general ledger application receives the request message and produces the response message that is sent back to the enterprise application. The general ledger application uses information in the request message to know how to send the response message back to the enterprise application. This scenario assumes that the general ledger component “owns” the chart of accounts definition and the instances of data within it.

A List LDGRACTUAL XML is used to publish one or more summary listings of ledger information. This may be in response to a Getlist LDGRACTUAL request or to published proactively as a listing of summary ledger information for a business event. This ensures that subsidiary ledgers which keep details of the financial transactions are reflected correctly in summary in the general ledger application. Ledger actuals may also be used in the analysis of accounts.

Get LDGRACTUAL and Show LDGRACTUAL

Get LDGRACTUAL XML enables an enterprise application to request actual detailed accounting data from a general ledger system. General Ledger is a central repository of accounting data that is summarized into account balances for analysis and reporting in the books. Each subsidiary ledger usually sends journal data (usually in summary) to the general ledger. The general ledger system then posts this details to the appropriate accounts and books. The Get LDGRACTUAL XML is used to request specific ledger information.

The purpose of the Show LDGRACTUAL XML is to communicate to an enterprise application (subsidiary ledgers) the sending systems representation of ledger information specifically requested. This may be used as a to a Get LDGRACTUALS XML or as a push notification of an event (updated general ledger data). This XML may be used individually or as part of a larger interface scenario. This XML uses a Data Type called LDGRACTUAL, which represents any of the general ledger elements corresponding to the Account Structure hierarchy that can be queried.

Acknowledge Delivery

A DELIVERY is an XML business document that details the receipt of goods by a customer or consignee destination from a supplier business partner using the services of a transportation provider (carrier). The DELIVERY document has been modeled for use in conjunction with the SHIPMENT XML when issued by the shipping business partner as an Advance Shipment Notification. The environment for this XML can be within the enterprise or outside the enterprise. DELIVERY was designed to support the physical process of receiving of goods from a carrier and is not biased toward receiving against a particular purchase, transfer or sales order reference document. This feature enables the receiving of goods simultaneously across multiple orders that were consolidated for shipping efficiency.

The Acknowledge Delivery XML may be used individually, or as part of a larger integration scenario. The ACKNOWLEDGE DELIVERY document may be used to notify the shipping business partner that the shipment has been received by the customer or consignee destination, and alert them to any discovered discrepancies. The acknowledgement may contain the full detail of the receipt as created by the receiving party or just the discrepancies and other exception conditions. This XML supports receipt acknowledgements at either the line item level and/or the ship unit level. Intermediate transportation or logistics providers or freight forwarding partners can use this document to acknowledge the receipt of entire shipping units without detailing the corresponding contents.

Receive Delivery

A DELIVERY is an XML business document that details the receipt of goods by a customer or consignee destination from a supplier business partner using the services of a transportation provider (carrier). The DELIVERY document has been modeled for use in conjunction with the SHIPMENT XML when issued by the shipping business partner as an Advance Shipment Notification. The environment for this XML can be within the enterprise or outside the enterprise. DELIVERY was designed to support the physical process of receiving of goods from a carrier and is not biased toward receiving against a particular purchase, transfer or sales order reference document. This feature enables the receiving of goods simultaneously across multiple orders that were consolidated for shipping efficiency.

The Receive Delivery XML may be used individually, or as part of a larger integration scenario. The RECEIVE DELIVERY document may be used to update the receiver's internal receiving and order management business applications to indicate that the requested material has arrived, including any unexpected quantity, condition or other exception discrepancies. This XML supports receiving at either the line item level and/or the ship unit level. Intermediate transportation or logistics providers or freight forwarding partners can use this document to acknowledge the receipt of entire shipping units without detailing the corresponding contents.

Get Delivery and Show Delivery

A DELIVERY is an XML business document that details the receipt of goods by a customer or consignee destination from a supplier business partner using the services of a transportation provider (carrier). The DELIVERY document has been modeled for use in conjunction with the SHIPMENT XML when issued by the shipping business partner as an Advance Shipment Notification. The environment for this XML can be within the enterprise or outside the enterprise. DELIVERY was designed to support the physical process of receiving of goods from a carrier and is not biased toward receiving against a particular purchase, transfer or sales order reference document. This feature enables the receiving of goods simultaneously across multiple orders that were consolidated for shipping efficiency.

The Get Delivery XML may be used individually, or as part of a larger integration scenario. For example, a Warehouse Management system may send a Get Delivery XML to an Order Management system. The GET DELIVERY XML may be used to request information about a specific expected (unreceived) or previously received goods delivery.

The Show Delivery is an XML document that may be used to obtain information about a specific expected (unreceived) or previously received goods delivery. The SHOW DELIVERY XML supports describing shipment content at either the line item level and/or the ship unit level. Intermediate transportation or logistics providers or freight forwarding partners can use this document to acknowledge the receipt of entire shipping units without detailing the corresponding contents. The Show Delivery XML may be issued in response to a Get Delivery request, or emitted asynchronously for notification upon some business event.

For expected deliveries, the SHOW DELIVERY document content may act as a receiving template or checklist to identify the quantity and shipping configuration of the expected goods. In this manner the SHOW DELIVERY XML may be considered logically equivalent to the Advance Ship Notice information in a SHOW SHIPMENT document. This similarity is by design, as the receiver's SHOW DELIVERY may be directly derived from shipper's SHOW SHIPMENT content after the receiver's business logic has appropriately validated the Advance Ship Notification information.

Getlist Delivery and List Delivery

A DELIVERY is an XML business document that details the receipt of goods by a customer or consignee destination from a supplier business partner using the services of a transportation provider (carrier). The DELIVERY document has been modeled for use in conjunction with the SHIPMENT XML when issued by the shipping business partner as an Advance Shipment Notification. The environment for this XML can be within the enterprise or outside the enterprise. DELIVERY was designed to support the physical process of receiving of goods from a carrier and is not biased toward receiving against a particular purchase, transfer or sales order reference document. This feature enables the receiving of goods simultaneously across multiple orders that were consolidated for shipping efficiency.

The GETLIST DELIVERY is an XML that is used to request information about a set of expected (unreceived) or previously received goods deliveries meeting certain selection criteria. The response to the GETLIST DELIVERY request is LIST DELIVERY XML. The GetList Delivery XML may be used individually, or as part of a larger integration scenario. For example, a Warehouse Management system may send a Getlist Delivery XML to an Order Management system to request information on previously received goods deliveries.

The LIST DELIVERY is an XML document used to obtain limited information listing about an expected (unreceived) or previously received goods deliveries that match certain selection criteria in a Getlist Delivery request. Additional information about a specific DELIVERY may be obtained through a Show Delivery XML by using the listing information to populate a GET DELIVERY request. The XML provides general information about the delivery receipt document, the partner including the address and contacts, information on the item inventory being delivered, information on the particular shipping unit being received, and specific quantity of goods contained within a shipping unit. The List Delivery XML returns specific data that matches the selection criteria provided in a Getlist Delivery XML.