BEA WebLogic Adapter for SAP User's Guide
Version 5.5.011
For WebLogic Server 9.1

DN3501342.0306
March 14, 2006

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 © 2006, by BEA Systems, Inc. All rights reserved. Patent Pending. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Preface

This document is written for system integrators who develop client interfaces between SAP and other applications. It describes how to use the BEA WebLogic Adapter for SAP to integrate SAP IDocs, RFCs, and BAPIs with your BEA WebLogic Server. It is assumed that readers understand Web technologies and have a general understanding of Microsoft Windows and UNIX systems.

How This Manual Is Organized

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

Chapter/Appendix		Contents
1	Introducing the BEA WebLogic Adapter for SAP	Provides an overview of the BEA WebLogic Adapter for SAP. Discusses key features and functionality of the adapter.
2	Configuring SAP Inbound Processing	Describes how to configure your SAP system for inbound (client) processing.
3	Creating XML Schemas for SAP	Describes how to create XML schemas for SAP business objects using Servlet Application Explorer.
4	Creating and Publishing Integration Business Services	Describes how to create and publish Integration Business Services using Servlet Application Explorer.
5	Configuring the Event Adapter for SAP	Describes how to create ports and channels to listen for SAP events using Servlet Application Explorer.
6	Management and Monitoring	Describes how to use the management and monitoring tools provided by the Integration Business Services Engine and the JCA Test Tool.
7	Understanding SAP Events	Describes how to configure and test your SAP system for event processing.

Chapter/Appendix		Contents
8	Troubleshooting and Error Messages	Describes limitations and workarounds when connecting to SAP. The adapter-specific errors listed in this chapter can arise whether using the adapter with a JCA, or with an iBSE configuration.
A	Sample Files and Coding Techniques	Provides sample request and response documents sent between SAP and the BEA WebLogic Adapter for SAP as well as a sample RFC module.
B	Best Practices and Performance Considerations	Provides best practices and performance considerations for the BEA WebLogic Adapter for SAP when deployed to the BEA WebLogic Server.

Documentation Conventions

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

Convention	Description
`THIS TYPEFACE` or `this typeface`	Denotes syntax that you must enter exactly as shown.
`this typeface`	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
`underscore`	Indicates a default setting.
this typeface	Represents a placeholder (or variable), a cross-reference, or an important term.
this typeface	Highlights a file name or command.
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 (…).

Convention	Description
. . .	Indicates that there are (or could be) intervening or additional commands.

Contact Us!

Your feedback on the BEA WebLogic Adapter for SAP 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 SAP documentation.

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

If you have any questions about this version of BEA WebLogic Adapter for SAP, or if you have problems using the BEA WebLogic Adapter for SAP, 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, or 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 *not* 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
Trace and log files
Log transaction

Contents

1. Introducing the BEA WebLogic Adapter for SAP 1-1

Features of the BEA WebLogic Adapter for SAP 1-2
SAP Certification 1-2
Supported Versions and Platforms 1-3
SAP Business Components 1-4
Integrating With SAP 1-5
Component Information for the BEA WebLogic Adapter for SAP 1-6
Component Information Roadmap 1-6
Application Explorer 1-6
Integration Business Services Engine 1-7
Enterprise Connector for J2EE Connector Architecture (JCA) 1-7
Web Services and Java Connector Architecture Functionality 1-8
Web Services 1-8
Java Connector Architecture 1-8

2. Configuring SAP Inbound Processing 2-1

Overview 2-2
Configuring a Logical System 2-3
Configuring a Distribution Model 2-6
Defining a Partner Profile 2-9

3. Creating XML Schemas for SAP 3-1

Overview 3-2
Starting Servlet Application Explorer 3-2
Establishing a Target for SAP 3-3
Creating a New Target 3-3
Connecting to a Target 3-11
Disconnecting From a Target 3-14
Modifying a Target 3-15
Deleting a Target 3-16
Viewing Application System Objects 3-16
Creating an XML Schema 3-20

4. Creating and Publishing Integration Business Services 4-1

Understanding Integration Business Services 4-2
Creating Integration Business Services 4-2
Creating Business Services With Application Explorer 4-2
Generating WSDL From a Web Service 4-11
Identity Propagation 4-24

5. Configuring the Event Adapter for SAP 5-1
Understanding Event Functionality 5-2
Creating, Editing, or Deleting a Port 5-2
Using the Default Event Port 5-16
Creating, Editing, or Deleting a Channel 5-16
Deploying Components in a Clustered BEA WebLogic Environment 5-26
Synchronous Event Processing 5-33

6. Management and Monitoring 6-1
Managing and Monitoring Services and Events Using iBSE 6-2
Managing and Monitoring Services and Events Using the JCA Test Tool 6-16
Setting Engine Log Levels 6-20
Configuring Connection Pool Sizes 6-22
Migrating Repositories 6-23
Exporting or Importing Targets 6-23
Retrieving or Updating Web Service Method Connection Information 6-28
Starting or Stopping a Channel Programmatically 6-32

7. Understanding SAP Events 7-1
Overview 7-2
Related Concepts and Terminology 7-2
Client and Server Programs 7-2
SAP Gateway 7-2
Program IDs and Load Balancing 7-3
Registering Your Program ID in SAPGUI 7-3
Testing the SAP Event Adapter 7-6
Application Link Enabling Configuration for the Event Adapter 7-8
Document Processing 7-8
Collected Inbound Configuration 7-8
Testing the SAP ALE Configuration 7-17
Usage Considerations 7-18
Multiple Events Using Identical Program IDs 7-18

8. Troubleshooting and Error Messages 8-1
Troubleshooting Overview 8-2
Error Messages in Application Explorer 8-2
Error Messages in SAP 8-5
Error Messages in JCA 8-6
Error Messages in iBSE 8-6
General Error Handling in iBSE 8-6
Adapter-Specific Error Handling 8-7

A. Sample Files and Coding Techniques A-1
Sample RFC Request Document A-2

Sample RFC Response Document A-2
Sample IDoc XML for Message Type DEBMAS A-3
Collected IDocs A-9
Sample RFC Module A-9
Sample Wrapper Module to Call Functions on Remote Destinations (Service) A-10
Using Staging BAPIs to Retrieve SAP BW Metadata A-12

B. Best Practices and Performance Considerations B-1

Overview of the BEA WebLogic Adapter for SAP B-2
Adapter Framework B-2
Supported Versions and Platforms B-2
Integrating With SAP B-3
Supported Tools and Technologies B-4
Application Explorer B-4
Integration Business Services Engine B-5
Enterprise Connector for J2EE Connector Architecture (JCA) B-5
Understanding Web Services and Java Connector Architecture Functionality B-6
Web Services B-6
Java Connector Architecture B-6
Understanding JCA Managed and Non-managed Modes B-6
JCA Managed Mode B-7
JCA Non-managed Mode B-8
Application Design B-8
Web Services and JCA Usage Considerations B-8
Configuring a Clustered Environment B-9
Synchronous Event Processing B-13
Configuration Tuning B-15
Repository Migration Using iBSE Administrative Services B-15
JCA Troubleshooting B-16

CHAPTER 1

Introducing the BEA WebLogic Adapter for SAP

Topics:

- Features of the BEA WebLogic Adapter for SAP
- SAP Certification
- SAP Business Components
- Integrating With SAP
- Component Information for the BEA WebLogic Adapter for SAP
- Web Services and Java Connector Architecture Functionality

The following section provides an overview of the BEA WebLogic Adapter for SAP.

Features of the BEA WebLogic Adapter for SAP

The BEA WebLogic Adapter for SAP is a remote function call adapter that provides a means to exchange real-time business data between SAP R/3 systems and other application, database, or external business partner systems. The adapter enables external applications for inbound and outbound processing with SAP.

The adapter uses XML messages to enable non-SAP applications to communicate and exchange transactions with SAP using one of the following two methods.

- **Event Adapter.** Applications use this capability if they require access to SAP data only when an SAP business event occurs.
- **Service Adapter.** Applications use this capability when they must initiate an SAP business event.

The BEA WebLogic Adapter for SAP provides:

- Support for bidirectional message interactions.
- The Servlet Application Explorer, a GUI tool that uses SAP object repository metadata to build XML schemas and Web services to handle adapter requests or event data.
- Support for Remote Function Calls (RFC), Business Application Programming Interfaces (BAPI), and Intermediate Documents (IDoc) interfaces to SAP.

SAP Certification

The BEA WebLogic Adapter for SAP provides state-of-the-art middleware solutions for SAP Basis and SAP Web application server-based systems. The adapter has earned the following three interface certifications which demonstrate that it promotes cost-effective and low-risk solutions.

- **CA-ALE certification.** Enhances electronic data interchange (EDI) subsystem interface with SAP Basis and SAP Web Application Server. Using direct program-to-program remote communication and transformation from non-SAP systems to SAP solution-based systems, the adapter expedites the conversion, import, and export of critical intermediate documents (IDocs).
- **CA-AMS certification.** Rapidly bridges SAP Basis and SAP Web Application Server data exchange with other applications through pure message delivery. As an ALE (Application Link Enabling) Message Handler, the adapter sends IDoc messages without a requirement for conversion from one or more SAP solution-based systems.

- **CA-XML certification.** Eases the communication between external middleware with SAP Basis and SAP Web Application Server over the Internet using XML, HTTP, or HTTPS. The adapter immediately transforms SAP solution specifications into XML for straight transfer into application subsystem repositories. The adapter directly receives and converts messages to be pulled or pushed into XML formats to or from SAP solution-based systems over the Internet.

Supported Versions and Platforms

The BEA WebLogic Adapter for SAP supports R/3 and R/3 Enterprise. The following SAP R/3 versions are supported by the BEA WebLogic Adapter for SAP:

- 4.6C
- 4.6D
- SAP R/3 Enterprise 47x100

  SAP R/3 Enterprise Version 47x100 is supported on the SAP Web Application Server Versions 6.20 and 6.40.
- SAP R/3 Enterprise 47x200

  SAP R/3 Enterprise Version 47x200 is supported on the SAP Web Application Server Versions 6.20 and 6.40.
- SAP Java Connector (SAP JCo) 2.16

  Note: For the current release status of the SAP Java Connector, refer to SAP Note #549268 in the SAP Service Marketplace.

In addition, the BEA WebLogic Adapter for SAP supports those SAP versions that are on the mainstream maintenance track as defined by the SAP Service Marketplace release strategy. To access this document, use the following URL:

`http://service.sap.com/releasestrategy/`

If the SAP R/3 version you are looking for is not listed in this table, please consult your BEA Customer Service Representative.

The following 32-bit platforms are supported by the BEA WebLogic Adapter for SAP:

- MS Windows 2000 with SP2, MS Windows 2003
- Sun Solaris 8 and Sun Solaris 9
- HP-UX 11
- IBM AIX 5
- Red Hat Enterprise Linux ES version 3.0
- TRU64

- OS/400
- Z/OS

For additional platform or operating system support information for the BEA WebLogic Adapter for SAP, contact Customer Support.

SAP Business Components

The BEA WebLogic Adapter for SAP is designed to provide standard access to SAP business components such as Remote Function Call (RFC) modules, BAPIs (Business Application Programming Interfaces), and IDocs (Intermediate Documents), that are used to support existing business processes.

These business components and methods are available to the adapter as requests of SAP and to the event adapter when SAP invokes its remote requests and work in the following ways:

Business Application Programming Interfaces (BAPIs) are interfaces within the business framework that are used to link SAP components to one another or to third-party components. BAPIs are called synchronously and return information.

Remote Function Call (RFC) Modules are SAP application interfaces that enable clients to invoke SAP technologies and receive responses.

Note: Depending on the release or service pack installed, certain RFCs may not exist in your particular SAP system. Therefore, the examples included in this documentation may not be relevant to your system. If this is the case, you should use the examples as a general reference for adapter functionality and choose an RFC that exists within your SAP application environment.

As described in SAP Release Note 109533, SAP Function Modules (RFCs) can be delivered with different release statuses. SAP supports only RFCs that are awarded with the Released for Customer status. There is no claim to the release independencies of the interfaces and the continued existence/functionality of the modules. For more information on the status of a specific function module, consult your SAP Service Marketplace.

Intermediate Documents (IDocs) are the "logical messages" that correspond to different business processes. They enable different application systems to be linked by a message-based interface. The IDoc type indicates the SAP format to use to transfer the data for a business transaction. An IDoc is a real business process in the form of an IDoc type that can transfer several message types. An IDoc type is described by the following components:

- **Control records.** A control record contains data that identifies the sender, the receiver, and the IDoc structure. An IDoc contains one control record.

- **Data records.** A data record consists of a fixed administration part and a data part (segment). The number and format of the segments can be different for each IDoc type.
- **Status records.** A status record describes the processing stages through which an IDoc passes.

  The following scenario is an example of IDoc functionality and its components:

  Purchase order number 4711 was sent to a vendor as IDoc number 0815. IDoc number 0815 is formatted in IDoc type ORDERS01 and has the status records "created" and "sent." The purchase order corresponds to the "logical" message ORDERS.

Integrating With SAP

You can use the BEA WebLogic Adapter for SAP to invoke an SAP business process, for example, add/update account, or you can use the adapter as part of an integration effort to connect SAP and non-SAP systems.

BAPIs and RFCs are called synchronously by the adapter and always return data (either technical error information or a well-formed response document). IDocs are processed asynchronously.

The adapter is bidirectional and can process an event in SAP by receiving RFCs and IDocs directly from SAP. The SAP system can be configured to send an IDoc or RFC to a logical system when a certain event occurs, in this case to the adapter. The output sent by SAP can be in any of the following forms:

- An RFC request, for example, RFC_SYSTEM_INFO.
- A BAPI request, for example, BAPI_COMPANYCODE_GETLIST.
- An IDoc.

For request processing, the BEA WebLogic Adapter for SAP can send requests to SAP using the BAPI, RFC, or IDoc interfaces.

The adapter quickly and easily integrates your SAP IDocs, RFCs, and BAPIs with mission critical SAP system applications and other enterprise applications. The benefits of the adapter include:

- Elimination of the requirement for custom coding.
- Consistent data representation.

  Provides a standard XML representation of event data and request/response documents for SAP.

  The developer is freed from the specific details of the SAP interface (BAPI, RFC, IDoc) and the specific configuration details of the target SAP system.

- Adherence to SAP ABAP serialization rules and SAP Interface Repository standards published by SAP AG.

Component Information for the BEA WebLogic Adapter for SAP

The BEA WebLogic Adapter for SAP works in conjunction with one of the following components:

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

When the adapter is hosted in a third party application server environment, Application Explorer (used to configure SAP connections, create Web services, and configure event capabilities) can be configured to work in a Web services environment in conjunction with iBSE or JCA. When working in a JCA environment, the connector uses the Common Client Interface (CCI) to provide fast integration services using adapters instead of using Web services.

Component Information Roadmap

The following table lists the deployment component and the location of component information for the BEA WebLogic Adapter for SAP.

Deployed Component	**For more information, see**
Application Explorer	Chapters 3, 4, 5, and 6 of this guide *BEA WebLogic ERP Adapter Installation and Configuration*
Integration Business Services Engine (iBSE)	*BEA WebLogic ERP Adapter Installation and Configuration*
Enterprise Connector for J2EE Connector Architecture (JCA)	*BEA WebLogic ERP Adapter Installation and Configuration*

Application Explorer

Application Explorer uses an explorer metaphor to browse the SAP system for metadata. The explorer enables you to create XML schemas and Web services for the associated object. In addition, you can create ports and channels to listen for events in SAP. External applications that access SAP through the BEA WebLogic Adapter for SAP use either XML schemas or Web services to pass data between the external application and the adapter.

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, 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.

Enterprise Connector for J2EE Connector Architecture (JCA)

The Enterprise Connector for J2EE Connector Architecture (JCA) enables developers of JCA-compliant applications to deploy adapters as JCA resources. The connector is supported on J2EE-compliant application servers, such as your BEA WebLogic Server.

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.

Web Services and Java Connector Architecture Functionality

The following topics describe how the BEA WebLogic Adapter for SAP can incorporate Web services and Java Connector Architecture technology.

Web Services

Web services enable SAP calls to be made across the Internet or an intranet, using specialized versions of the XML language that allow a developer to specify the parameters, connections methods, and remote calls and store them for reference in a repository. At run time, a person, an interface, or another function, can read this repository and automatically invoke the service. Web services currently do not have industry standards for transactional behavior. Web services are useful when your function calls must be made across firewall boundaries. Using Web services, you can use functions provided by external providers, assuming you know the function interface.

Web Services Example:

A Web service exposes the "cup" interface, which provides a teacup. The Acme Company exposes the "tea" Web service, which provides a brown liquid when the correct parameter "money" is provided. A cup of tea can be received by invoking the "tea" Web service and passing the "money" parameter. Additional components are not required to receive tea using the "tea" Web service.

Java Connector Architecture

Java Connector Architecture (JCA) provides a reusable component model to build and deploy multi-tier applications that are platform and vendor independent. JCA acts as a type of envelope or "container" that enables the adapter to run inside the BEA WebLogic Server and connect to SAP and immediately return the results. JCA is useful when your SAP system resides within a local intranet or is accessed directly. JCA implements JAVA Connection and Transaction models. JCA requires a resource adapter to be physically deployed on the host application server to access the remote EIS system.

For more information on event capabilities of the JCA connectors, see Chapter 5, *Configuring the Event Adapter for SAP*. For more information on installing and deploying both connectors, see *BEA WebLogic ERP Adapter Installation and Configuration*.

Using combinations of JCA and Web services is possible. For example, a JCA application can be invoked by a Web service or a Web service may be implemented inside a JCA container. The standards and protocols are still evolving.

JCA Example

An application server, "Table" implements the "tea" container, which contains "tea, gas heat, water, and cup". A JCA container can be deployed that contains tea in the "Table" application server. If any of the elements of the "tea" container are missing, for example, "gas heat," then no tea is produced. The container must physically possess the "tea, gas heat, water, and cup" components to receive the tea.

CHAPTER 2

Configuring SAP Inbound Processing

Topics:

- Overview
- Configuring a Logical System
- Configuring a Distribution Model
- Defining a Partner Profile

The following section describes how to configure your SAP system for inbound (client) processing.

Overview

SAP Remote Function calls require no system setup other than the Connection Target Parameters. If you do not intend to send IDocs to the SAP system, you may skip this chapter.

In inbound IDoc processing, the adapter reads an XML document and creates an SAP Standard format IDoc. The adapter can optionally transform incoming data into the standard format through XML transformation or SAP ALE transformation. After assembled as an IDoc, the file is sent to SAP for inbound processing. Multiple IDocs can also be assembled into a larger document for efficient processing. For more information, see the SAP documentation.

ALE IDocs used for transmission of Electronic Data Interchange (EDI) messages require information about the intended target and method of transmission stored on the BEA WebLogic Server.

An IDoc consists of a Header that contains sender, processing type, receiver and other information, and multiple data segments, which contain the information to be processed. To store the parameters for processing of the IDoc messages, SAP requires a "logical system" entry for each transmission system. The Logical System stores type information about the Partner and the kind of messages expected from the partner.

The Partner Profile defines the kind of message and the type of SAP function called to process that particular IDoc in an application. SAP also requires a filtering model, whether or not it is used for a particular message, called a Distribution Model that defines a message type and applies optional segment filters. After these are established, Inbound ALE/IDoc processing can begin.

The Control Section or reference structure file EDI_DC40 (defined in SAP R/3), must be completed and contains all the identifying information about the IDoc. The assembled header and data records are sent to SAP by the adapter. SAP does not require an incoming port to be specified. It takes the incoming RFC stream and assigns a port designation. In the Partner Profile, a function module must be identified to process the IDoc in the R/3 system.

Usually, IDocs are written directly to the database and slowly read by the application (for example, Purchasing for Purchase Orders). This can take time depending on the type of data and the application. The adapter can "post to the database and return" or "post and wait." This is defined in the Partner Profile on the host system. In either case, you may send a status IDoc message to obtain the status of your IDoc or use appropriate transaction codes in SAP to view the IDocs online.

You must perform the following steps to configure SAP for inbound IDoc processing:

1. Configure a logical system.
2. Configure a distribution model.
3. Define an inbound partner profile.

Configuring a Logical System

In a distributed environment, each participating system must have a unique ID to avoid confusion. In SAP, the name of the logical system is used as the unique ID. This name is assigned explicitly to one client in an SAP system.

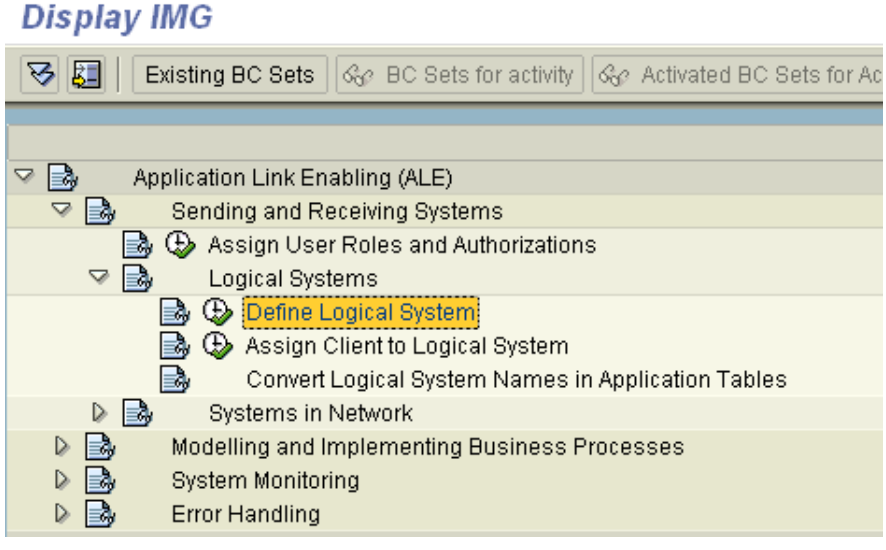
Procedure: How to Configure a Logical System

The following image shows the /nsale transaction in the field under the menu bar.

To configure a logical system:

1. Execute the *sale* transaction.

   The Display IMG window opens as shown in the following image.

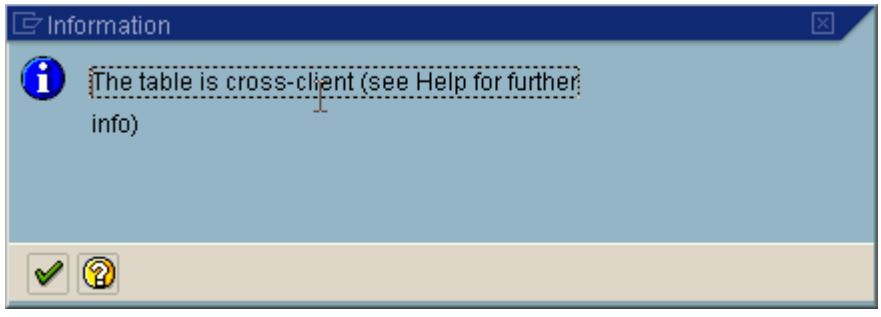


   a. Expand *Sending and Receiving Systems* and then, *Logical Systems*.

   b. Select *Define Logical System*.

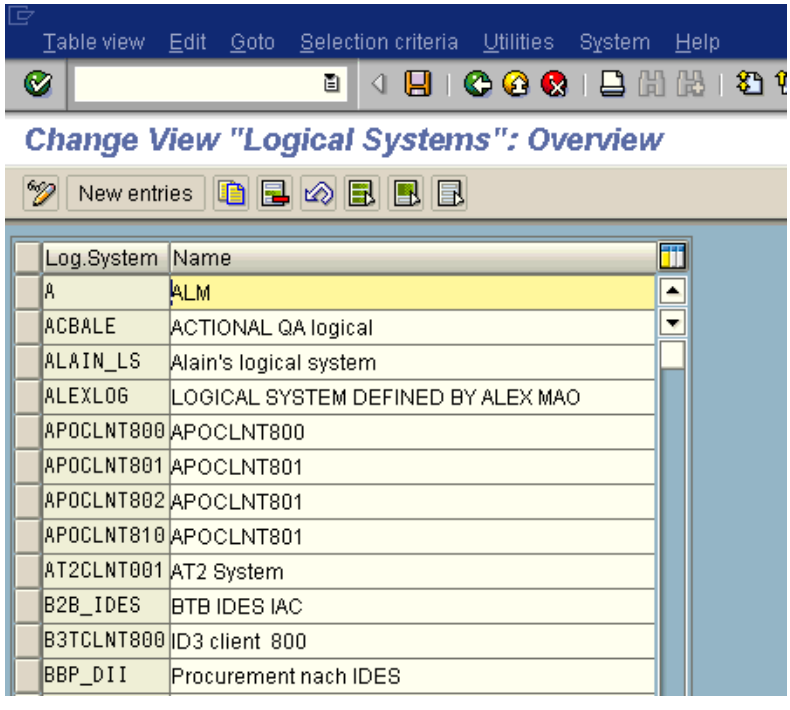
2. Click the *IMG - Activity* icon.

An information window appears that informs you that the table is cross-client as shown in the following image.



3. To continue, click the checkmark icon.

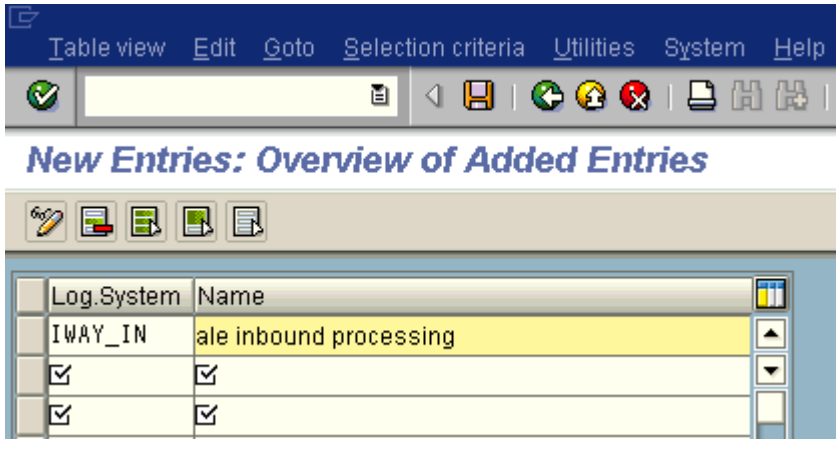
The Change View "Logical Systems" window opens with a list of logical systems and their names as shown in the following image.



Log.System	Name
A	ALM
ACBALE	ACTIONAL QA logical
ALAIN_LS	Alain's logical system
ALEXLOG	LOGICAL SYSTEM DEFINED BY ALEX MAO
APOCLNT800	APOCLNT800
APOCLNT801	APOCLNT801
APOCLNT802	APOCLNT801
APOCLNT810	APOCLNT801
AT2CLNT001	AT2 System
B2B_IDES	BTB IDES IAC
B3TCLNT800	ID3 client 800
BBP_DII	Procurement nach IDES

4. Click the *New entries* button.

The New Entries window opens where you can type information for the logical system and its corresponding name as shown in the following image.

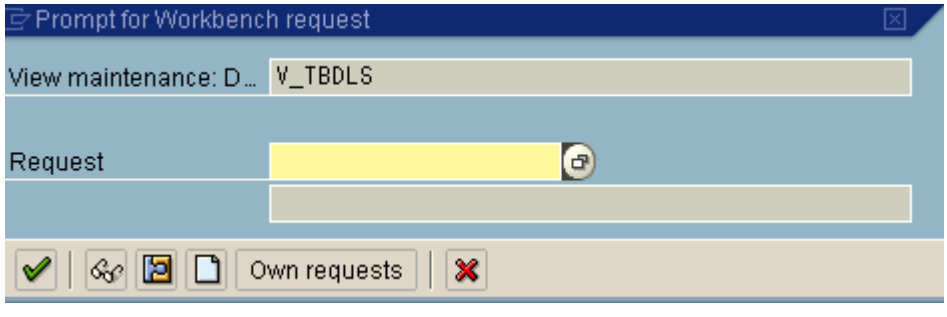


a. In the Log.System column, type the Logical System, for example, IWAY_IN.

b. In the Name column, type a corresponding description.

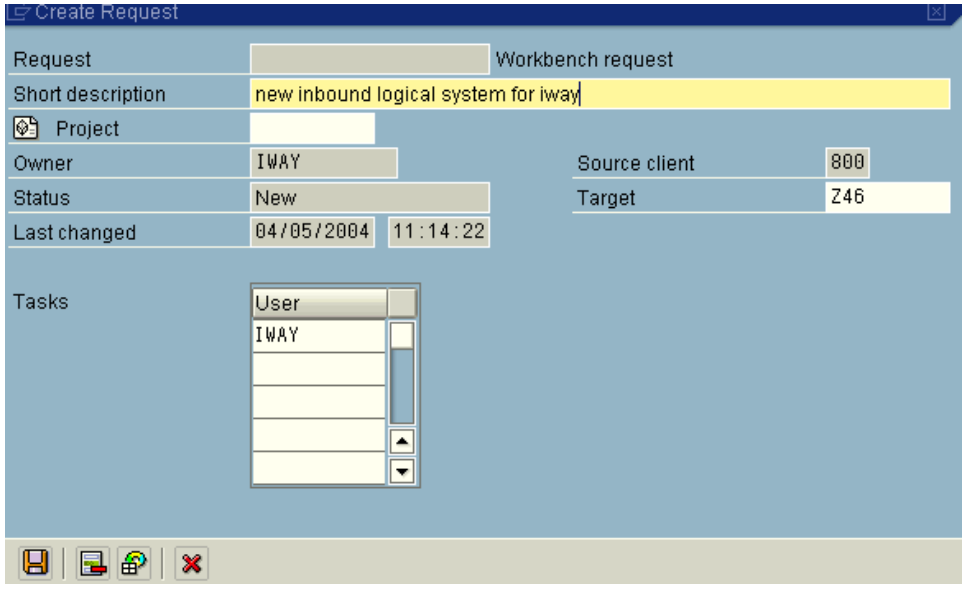
5. Click *Save*.

The Prompt for Workbench request window opens as shown in the following image. It includes fields for View maintenance and Request as well as several buttons.



6. Click the *Create Request* icon.

The Create Request window opens as shown in the following image. It includes fields that are already populated (such as Owner, Status, Last Changed, Source client, and so forth), empty fields (such as Request and Project) in which to specify information about your request, and a Tasks list.

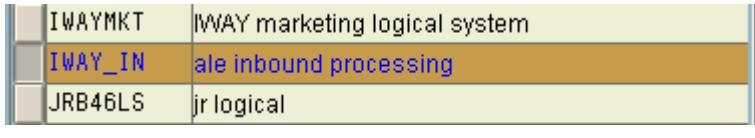


a. In the Request field, type a name.

b. In the Short description field, type a brief description of your request.

7. Click *Save*.

The logical system you configured, for example, IWAY_IN, appears in the list as shown in the following image.

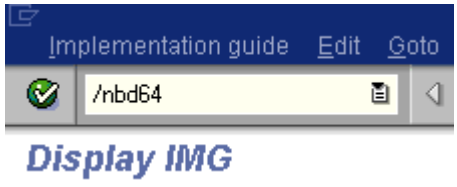


Configuring a Distribution Model

A distribution model is used to describe the ALE message flow between logical systems. Business objects are distributed to connected recipients according to a unique distribution model that can contain rules of varying complexity depending on the type of business objects involved.

Procedure: How to Configure a Distribution Model

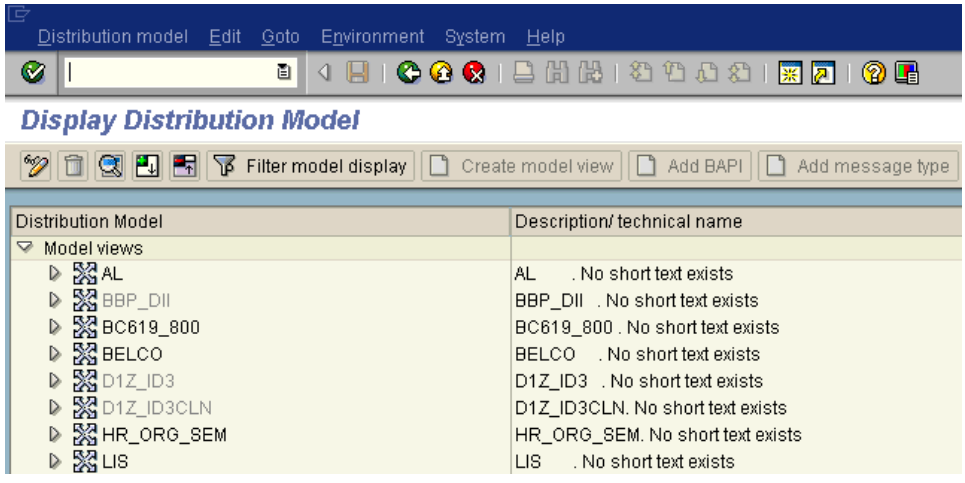
The following image shows the /nbd64 transaction in the field under the menu bar.



To configure a distribution model:

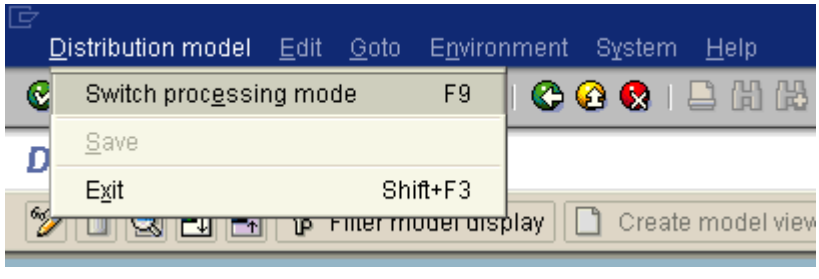
1. Execute the */bd64* transaction.

   The Display Distribution Model window opens and displays a list of available distribution models and their descriptions as shown in the following image.



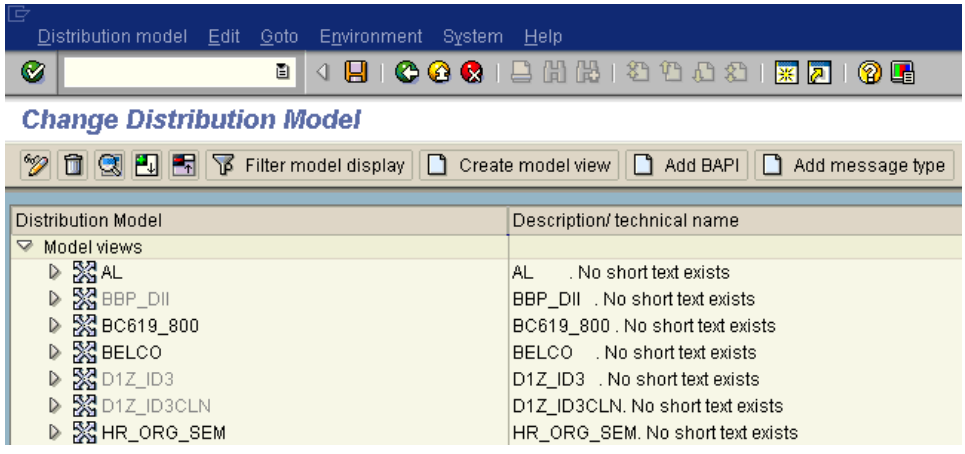
2. In the menu bar, click *Distribution model*.

   The Distribution model menu opens as shown in the following image.



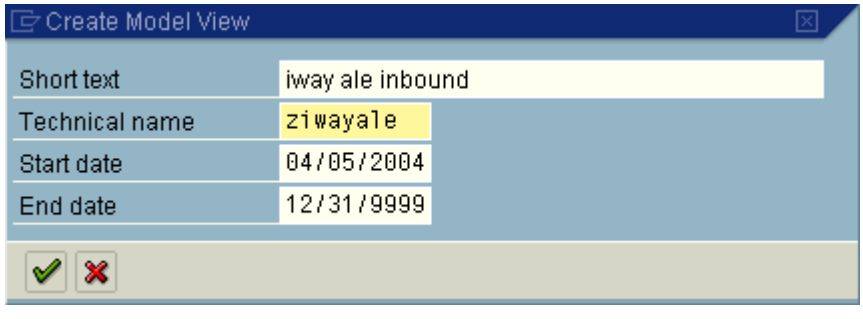
3. Select *Switch processing mode.*

The Display Distribution Model window switches to the Change Distribution Model. window as shown in the following image.



4. Click the *Create model view* button.

The Create Model View window opens and includes fields for the name of your distribution model and for Start and End dates as shown in the following image.



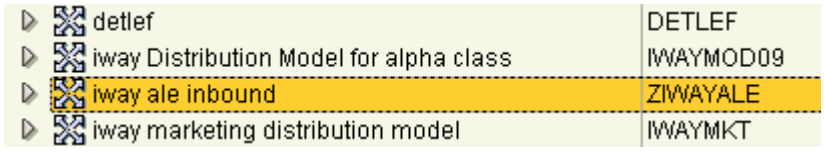
a. In the Short text field, type a model view name, for example, iway ale inbound.

b. In the Technical name filed, type a technical name, for example, ziwayale, which also serves as a description.

5. To enter the information, click the checkmark icon.

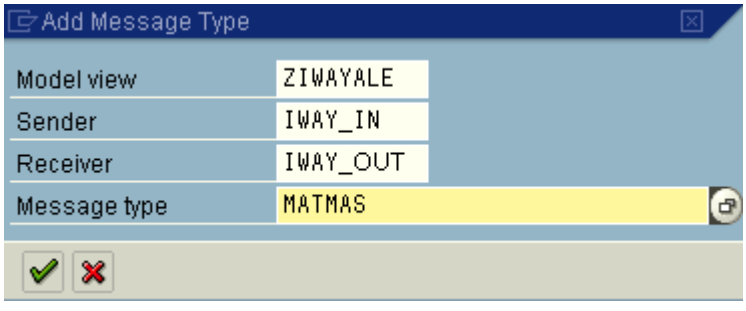
You are returned to the main Change Distribution Model window.

The distribution model you configured is now added to the list as shown in the following image.



6. Click the *Add message type* button.

   The Add Message Type window opens and includes fields where you can name and specify your message type.



   a. In the Sender field, type the logical system you configured, for example, IWAY_IN.

   b. In the Receiver field, type the logical system you configured, for example, IWAY_OUT.

   c. In the Message type field, type the message type to use, for example, MATMAS.

   To browse from a list of available message types, you can click the icon to the right of the field.

7. To enter the information, click the checkmark.

   You are returned to the main Change Distribution Model window.

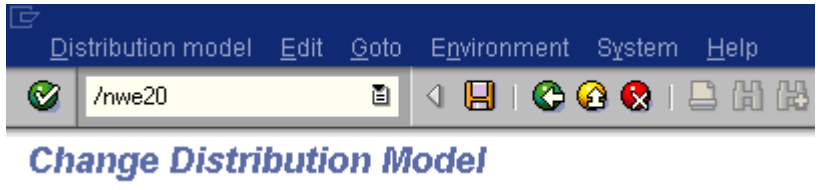
8. Click *Save*.

Defining a Partner Profile

Partner profiles are a requirement for data exchange. You define who can exchange messages with the SAP system using a specified port.

Procedure: How to Define a Partner Profile

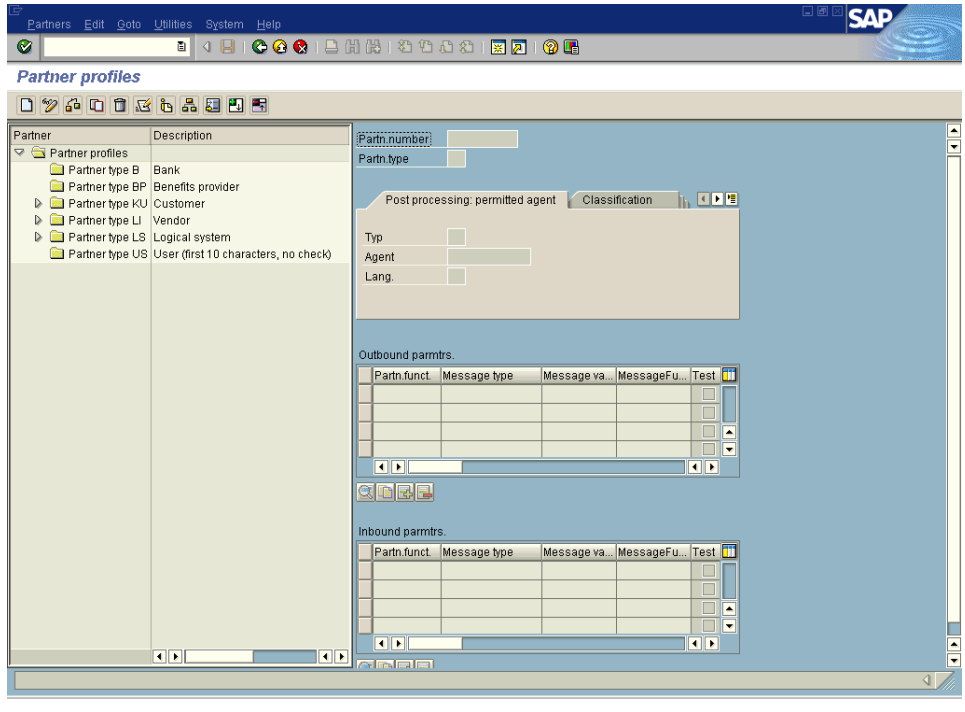
The following image shows the /nwe20 transaction in the field under the menu bar.



To define a partner profile for a specific IDoc:

1. Execute the *we20* transaction.

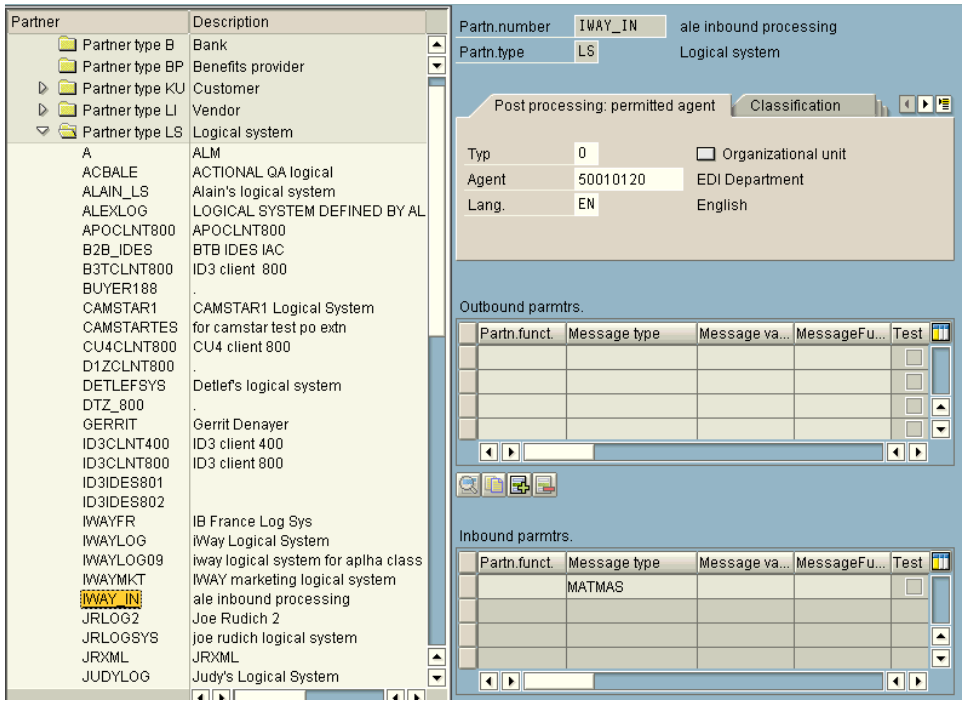
The Partner profiles window opens and displays two panes with information about the logical system as shown in the following image.



2. In the left pane, expand *Partner type LS* and select the logical system you configured from the list, for example, IWAY_IN.

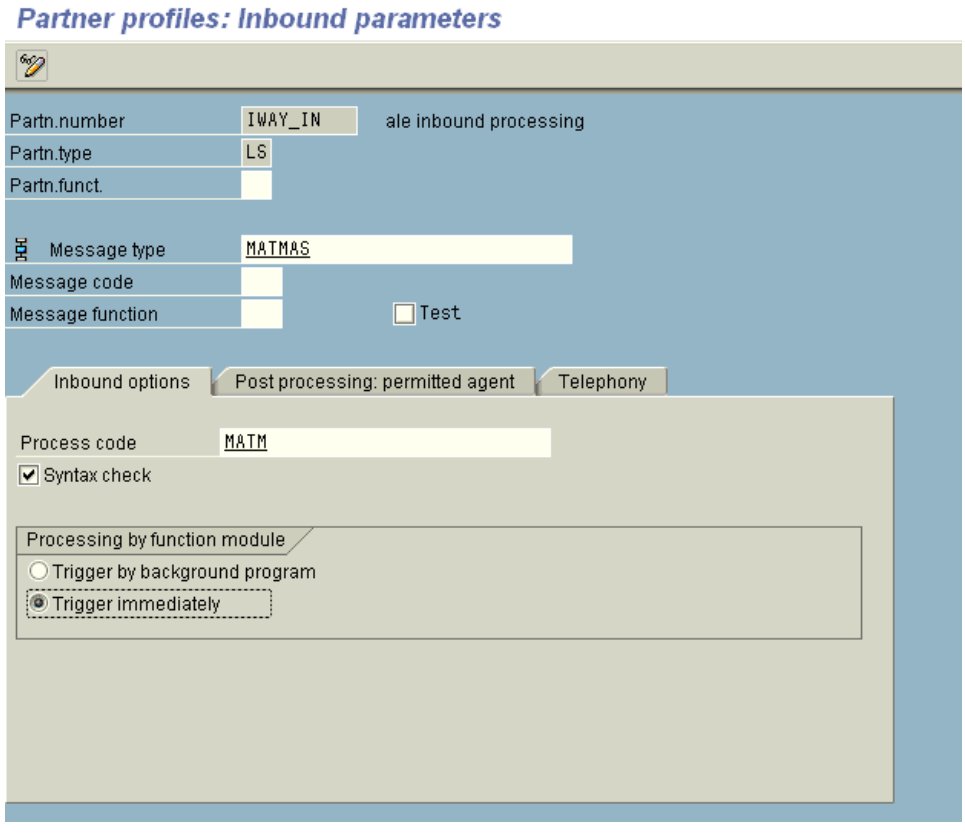
The right pane displays the details of the expanded folder including the logical system and type, language, and so forth, as shown in the following image.

Note: The Partn.number field refers to the name of the logical system.



3. Click *Save*.
4. From the Inbound parameters table in the lower right, click the *Create inbound parameter* icon.

The Partner profiles: Inbound parameters window opens as shown in the following image.



a. In the Message type field, type the message type to use, for example, MATMAS.

The Inbound options tab is selected by default.

b. In the Process code field, enter the process code you want to use, for example, MATM.

c. In the Processing by function module area, select one of the following options:

Trigger by background program. In this case, the BEA WebLogic Adapter for SAP writes IDocs to the SAP database, which are processed immediately.

Trigger immediately. In this case, the BEA WebLogic Adapter for SAP waits for the SAP system to process IDocs. This can take from one to fifteen minutes.

5. Click *Save*.

CHAPTER 3

Creating XML Schemas for SAP

Topics:

- Overview
- Starting Servlet Application Explorer
- Establishing a Target for SAP
- Viewing Application System Objects
- Creating an XML Schema

The following section describes how to create XML schemas for SAP business objects using Application Explorer.

The functionality of Application Explorer is standard for any deployment type. This section uses the Java™ servlet implementation of Application Explorer to provide examples.

Overview

The BEA WebLogic Adapter for SAP enables the processing of SAP BAPIs, RFCs, and IDocs.

External applications that access SAP through the adapter use either XML schemas or Web services to pass data between the external application and the adapter. You can use Servlet Application Explorer to create the required XML schemas and Web services.

SAP must be installed, configured, and available for client access. Application Explorer need not reside on the same system as the application system being accessed, but network access is required.

Starting Servlet Application Explorer

Before you can use the Servlet Application Explorer to browse metadata and create XML schemas, you must start the BEA WebLogic Server.

Procedure: How to Start BEA WebLogic Server on Windows

To start the BEA WebLogic Server on Windows:

1. Click the Windows Start menu.
2. Select *Programs*, *BEA Products,* and then click *WebLogic Server.*

Procedure: How to Start BEA WebLogic Server on UNIX

To start the BEA WebLogic Server on UNIX or from a command line:

1. Display a prompt.
2. At the prompt, type the following:

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

   where:

   BEA_HOME

   Is the directory where BEA WebLogic is installed.

   DOMAIN_NAME

   Is the domain you are using for iWay.

Procedure: How to Start Servlet Application Explorer

To start Application Explorer:

1. Enter the following URL in your browser window:

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

where:

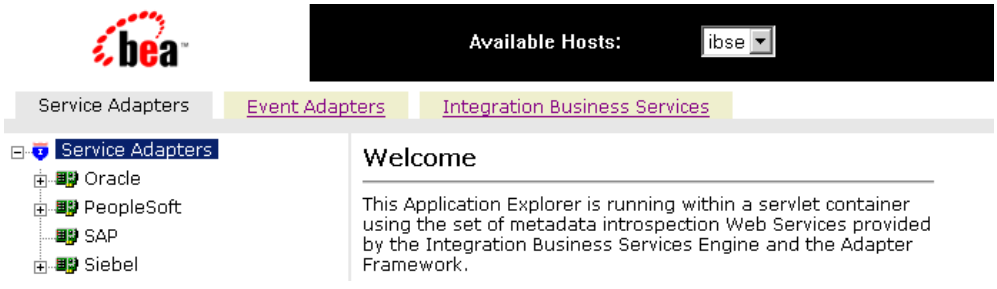
hostname

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

port

Is the HTTP port for the application server.

After you start Application Explorer, the following Welcome window opens, showing the Service Adapters, Event Adapters, and Integration Business Services tabs. The Service Adapters node is highlighted in the left pane.



The Available Hosts drop-down menu in the upper right lists the Connector for JCA or Servlet iBSE instance you can access.

2. For more information on adding instances, see *BEA WebLogic ERP Adapter Installation and Configuration*.

You are now ready to create new targets for SAP.

Establishing a Target for SAP

To browse SAP business objects, you must create a target for the system you intend to use. The target serves as your connection point and is automatically saved after you create it. You must establish a connection to this system every time you start Application Explorer or after you disconnect from the system.

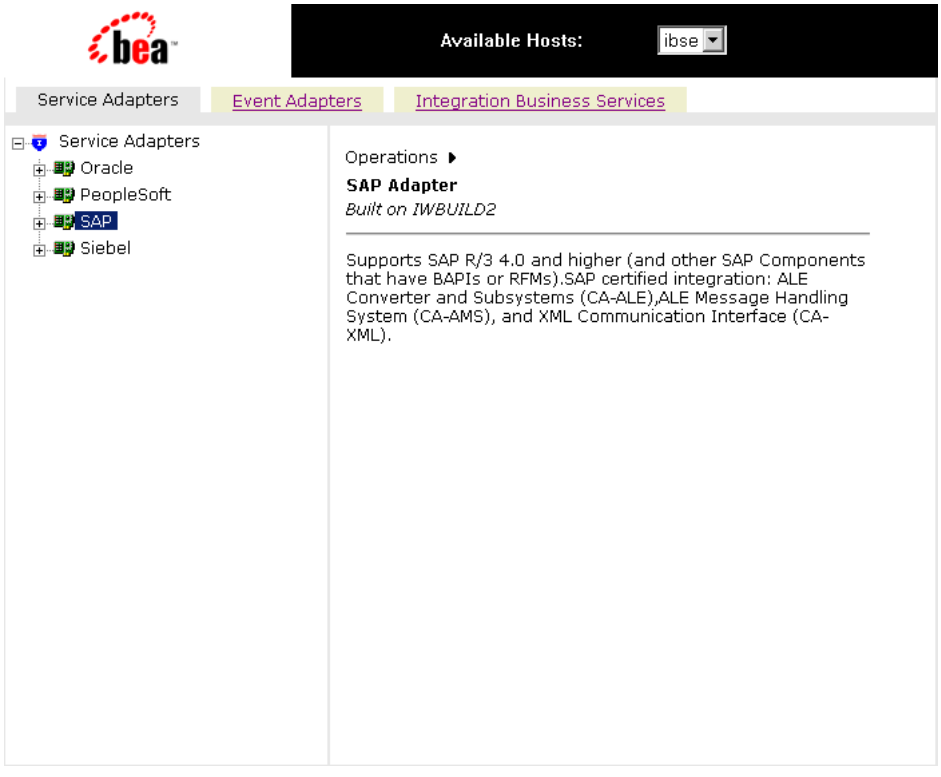
A list of supported application systems appears in the left pane of Application Explorer. The list is based on the adapters that you installed and have licenses to use.

Creating a New Target

To connect to an SAP system, you must define a new target. The target holds your logon parameters for the SAP system.

Procedure: How to Create a New Target

The following image shows a window with a navigation pane on the left that lists supported adapters. The right pane displays information about a selected adapter.



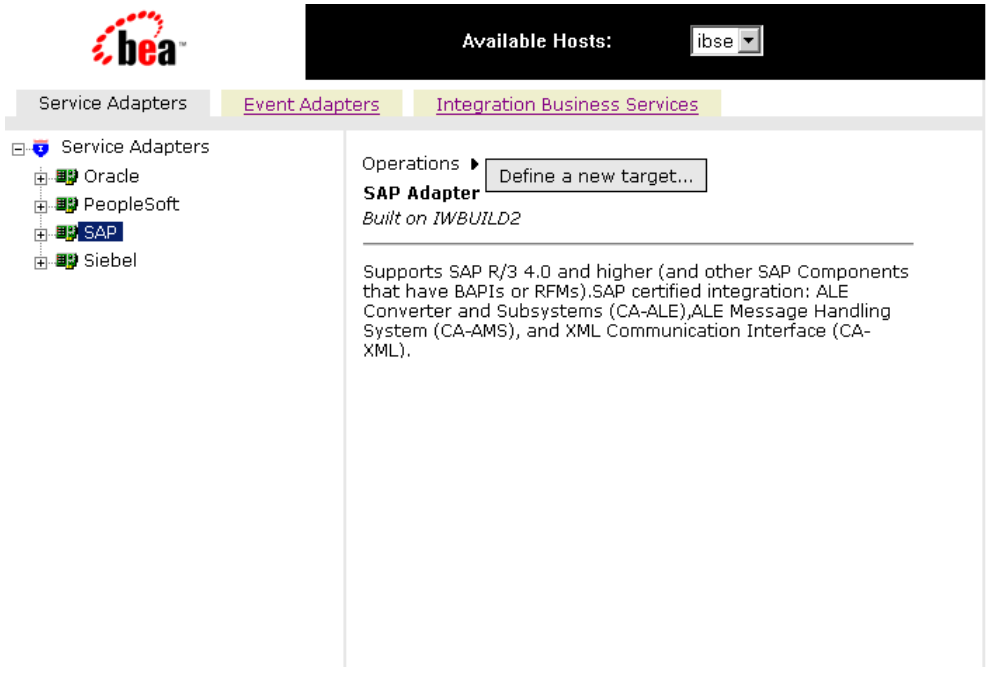
To create a new target:

1. In the left pane, click the *SAP* node.

   Descriptive information (for example, title and product version) about the BEA WebLogic Adapter for SAP appears in the right pane.

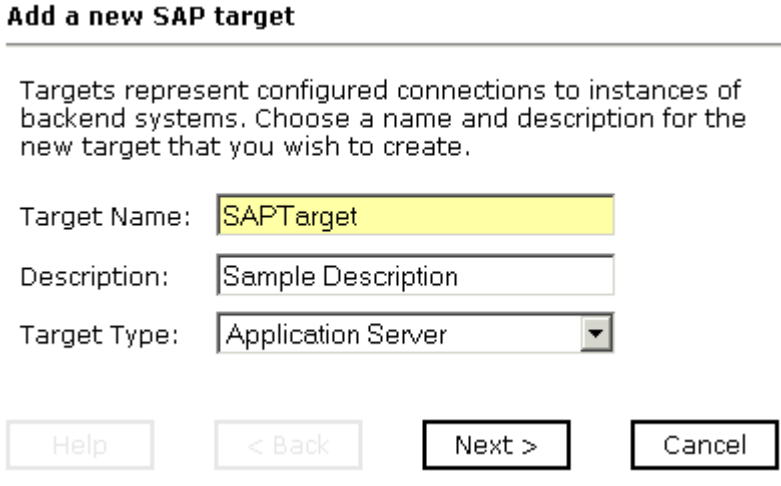
2. In the right pane, move the pointer over *Operations*.

The Define a new target menu option appears as well as title and product version information for the adapter in the right pane as shown in the following image.



3. Select *Define a new target*.

The Add a new SAP target pane opens on the right as shown in the following image.



a. In the Target Name field, type a name for the target, for example, SAPTarget.

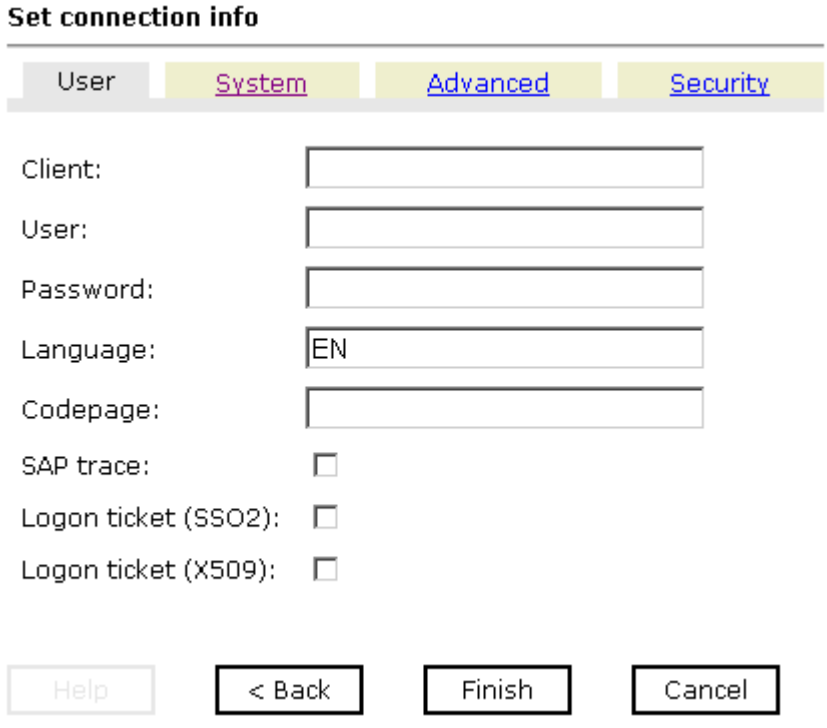
b. In the Target Description field, type a brief description (optional).

c. From the Target Type drop-down list, select the type of target to connect to.

The default value is Application Server.

4. Click *Next*.

The Set connection info pane opens on the right. The following tabs are available: User, System, Advanced, and Security. The User tab is active as shown in the following image.



The User (required) tab enables you to provide authentication information for the SAP system to which you are connecting.

a. In the Client field, type the client number defined for the SAP application for client communications.

b. In the User field, type a valid user ID for the SAP application.

c. In the Password field, type a valid password for the SAP application.

d. In the Language field type a language key.

EN (English) is the default.

e. In the Codepage field, type a character code page value.

f. To enable traces, select the *SAP trace* check box.

g. If you are using a Secure Network Communications (SNC) adapter with SAP, select the *Logon ticket (SSO2)* check box.

h. If you are using an SNC adapter with SAP, select the *Logon ticket (X509)* check box.

Secure Network Communications (SNC) provides protection for the communication links between the distributed components of an R/3 System. Using SNC, SAP R/3 can support products that adhere to the GSS-API Version 2 standard. SNC supports application level (end-to-end security), Smartcard authentication, and Single Sign-On (SSO).

Note: Depending on the SAP system release, logging on using SSO or X.509 certificates is supported.

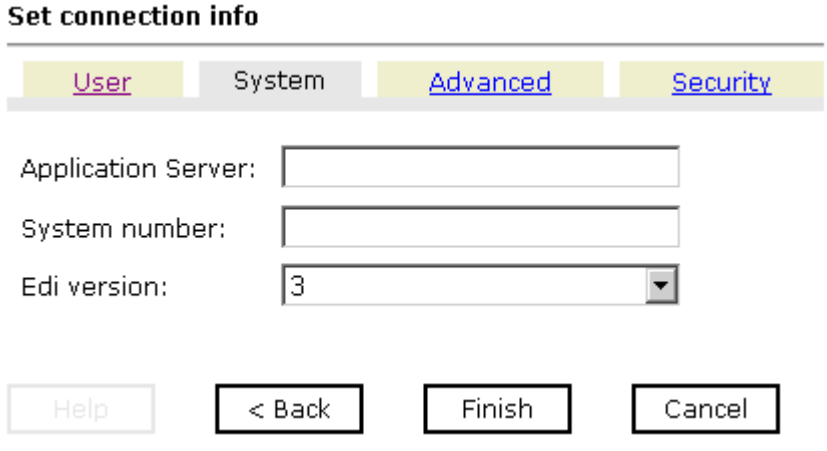
For **SSO**, specify the user to be *$MYSAPSSO2$* and pass the base64 encoded ticket as the passwd parameter.

For **X509**, specify the user to be *$X509CERT$* and pass the base64 encoded certificate as the passwd parameter.

For more information, see your SAP system documentation.

5. Click the *System* tab.

The System tab becomes available as shown in the following image.



The System (required) tab enables you to provide the application server name, system number, and EDI version for the SAP system to which you are connecting.

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

a. In the Application Server field, type the host name or IP address for the computer that is hosting the SAP application.

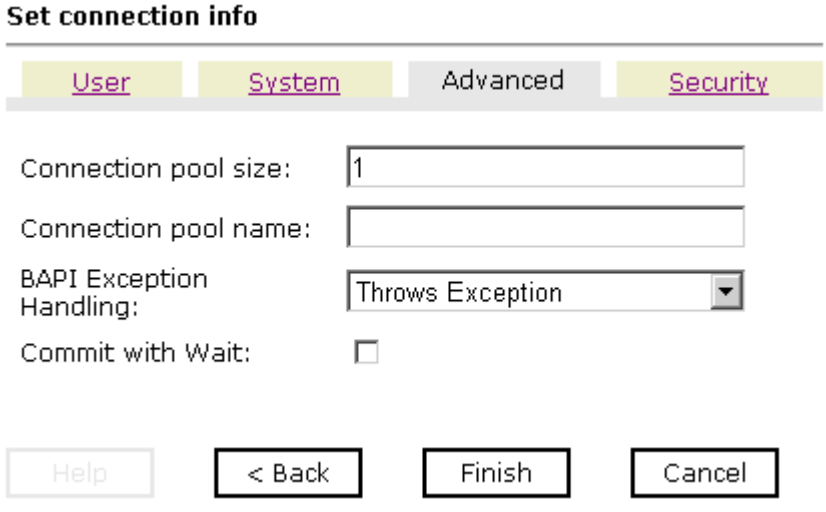
b. In the System Number field, type the system number defined to SAP for client communications.

c. From the EDI Version drop-down list, select the Electronic Data Interchange (EDI) document version you are using with the BEA WebLogic Adapter for SAP.

Version 3 is the default value.

6. To view the Advanced tab, click *Advanced*.

The Advanced tab becomes available as shown in the following image.



The Advanced tab enables you to specify SAP connection pooling information for the SAP system to which you are connecting and contains the following fields:

Connection pool size is used to specify the number of client connections in a pool you want to make available to SAP for Web service calls. A default connection pool size of 1 is available by default.

Connections to an R/3 server take up valuable resources on both the client and the remote server. You can create a pool of connections to minimize the resource and time constraints. In estimating the size of the pool, you may calculate pool size by the amount of server resources to be consumed, the number and size of the documents to be received, and the size of your Java Virtual Machine. The section of SAP documentation "Memory Management (BC-CST-MM)" explains in detail the resources required on the SAP system.

a. To use a connection pool, enter a value that is greater than 1.

The default value of 1 does not create a connection pool. Instead, a single SAP connection with sequential processing is shared. A pooled connection invokes multiple connections to SAP with parallel processing.

Note: If you are using Application Explorer to create Web services, the connection pool size value is used by your Web service during runtime. As a result, ensure that the connection pool size is sufficient for your purposes.

b. In the Connection pool name field, type a name for your connection pool only if you specified a connection pool size that is greater than 1.

c. From the BAPI Exception Handling drop-down list:

If your application is **Java centric**, select *Throws Exception* so that code components can catch the exception and react accordingly.

If your application is **document based**, select *Creates Document* to create an XML document that contains the Java exception.

It is up to your application to read the XML document and obtain the error.

d. If a high degree of accuracy is required in your application, select the *Commit with Wait* check box.

The adapter waits until all records are physically written to the database before returning from the function call. The "Commit With Wait" has a performance impact on adapter performance, so consider carefully before selecting it. The commit behavior of BAPIs is described in the SAP documentation under "BAPI Programming Guide and Reference (CA-BFA)."

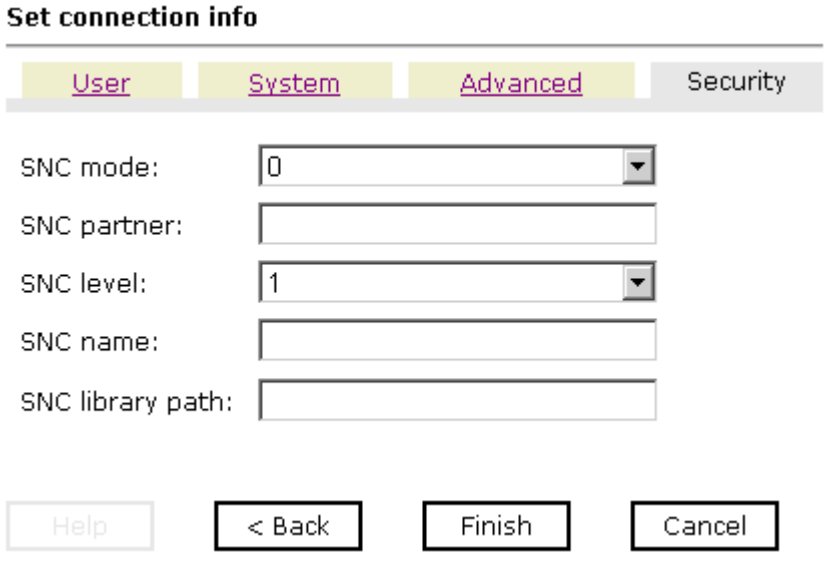
All SAP Business Objects that change data must commit work to the database. Some BAPIs developed in version 3.1 of the R/3 system use an internal commit behavior, and their commit behavior cannot be changed by the adapter. As soon as they are called, they commit the work they did.

BAPIs developed since release 3.1 use the external commit method. The adapter issues a commit command, and the commit is put in the database queue. If there is an application error in the first part of the commit, the error message "Posting could not be carried out" is returned, and the adapter rolls back the transaction. If in writing to the database, a database error occurs, a short dump is issued in the database records of SAP, but no message is returned to the adapter about the failure.

This option is disabled by default.

7. To view the Security tab, click *Security*.

The Security tab becomes active as shown in the following image.



The Security tab enables you to specify Secure Network Communication (SNC) information for the SAP system to which you are connecting.

SNC provides protection for the communication links between the distributed components of an R/3 System. Using SNC, SAP R/3 can support products that adhere to the GSS-API Version 2 standard. SNC supports application level (end-to-end security), Smartcard authentication, and SSO.

If you are using SAP Enterprise Portal, the J2EE engine generates the SAP logon ticket automatically. A possible SNC scenario would be from SAP Enterprise Portal to the BEA WebLogic Adapter for SAP.

If you want to use SAP logon tickets to enable SSO to non-SAP components, consult the SAP documentation regarding *Pluggable Authentication Services*. A possible SNC scenario in this case would be from a non-SAP Enterprise Portal to the BEA WebLogic Adapter for SAP.

a. To enable SNC, select *1* from the SNC mode drop-down list.

By default, SNC is disabled.

b. In the SNC partner field, type the name of the RFC server or message server (load balancing) that provides the SNC services.

c. From the SNC level drop-down list, select the version of the SNC library.

d. In the SNC name field, type the name of the SNC library you are using.

e. In the SNC library path field, type the path to the SNC library.

8. After you provide all the required information for your target, click *Finish*.

The SAP target (SAPTarget) appears below the sap node in the left pane as shown in the following image. You are now ready to connect to your SAP target.

Connecting to a Target

To connect to SAP, you use a target you defined, for example, the one in the previous procedure, SAPTarget.

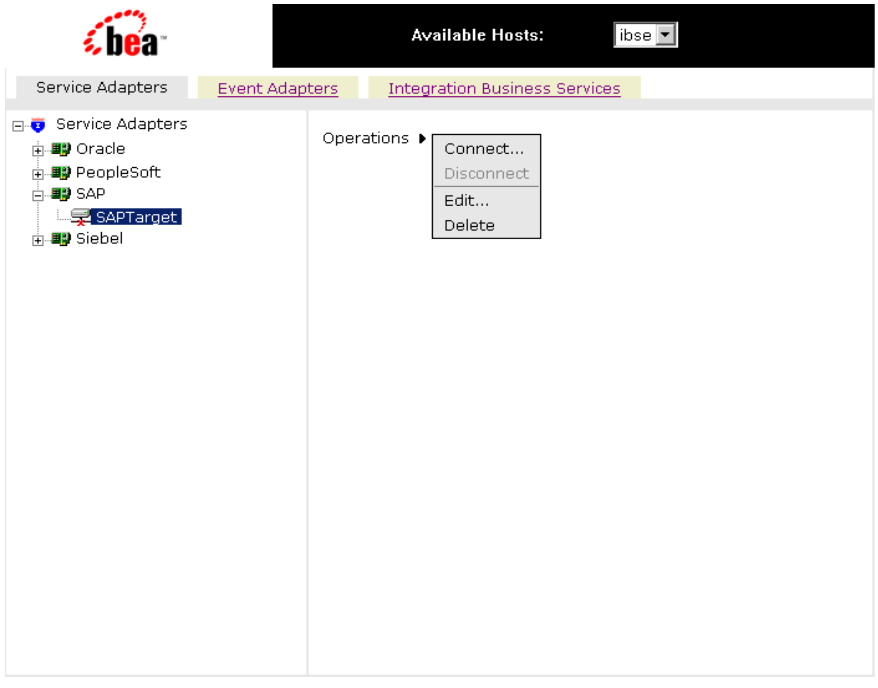
Procedure: How to Connect to a Target

To connect to a target:

1. In the left pane, expand the sap node and select the target you defined, for example, SAPTarget.

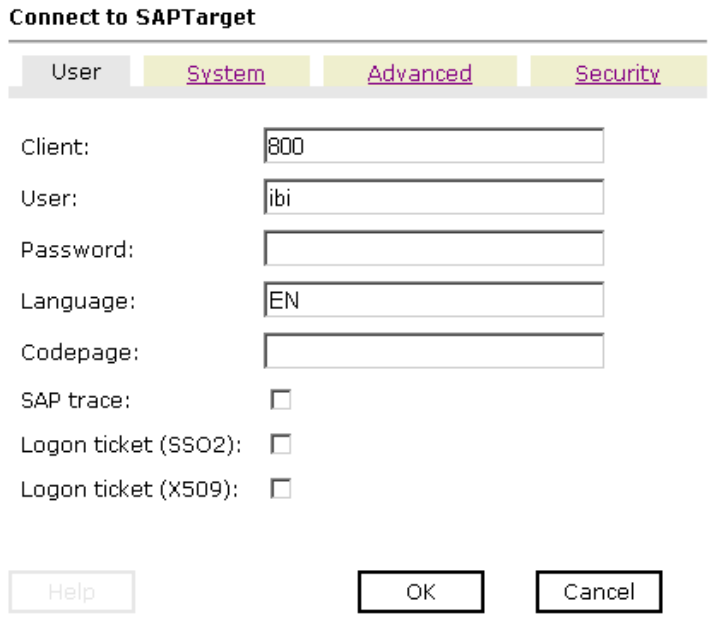
2. In the right pane, move the pointer over *Operations*.

The following image shows the target, with a red 'x', selected in the left pane. In the right pane, the Operations menu appears in its expanded form.



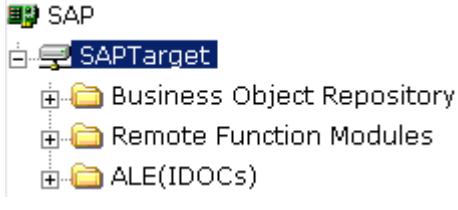
3. Select *Connect*.

The Connect to SAPTarget pane opens on the right, which contains fields with information for a client, user, and language, as well as fields for entering a password and a code page as shown in the following image. Additionally, you can select to enable an SAP trace and to enable security.



4. In the Password field, type a valid password and click *OK*.

The SAPTarget node in the left pane changes (the red 'x' disappears) to reflect that a connection was made as shown in the following image.



5. Expand the *SAPTarget* node.

The following SAP business objects appear:

- Business Object Repository
- Remote Function Modules
- ALE (IDocs)

Disconnecting From a Target

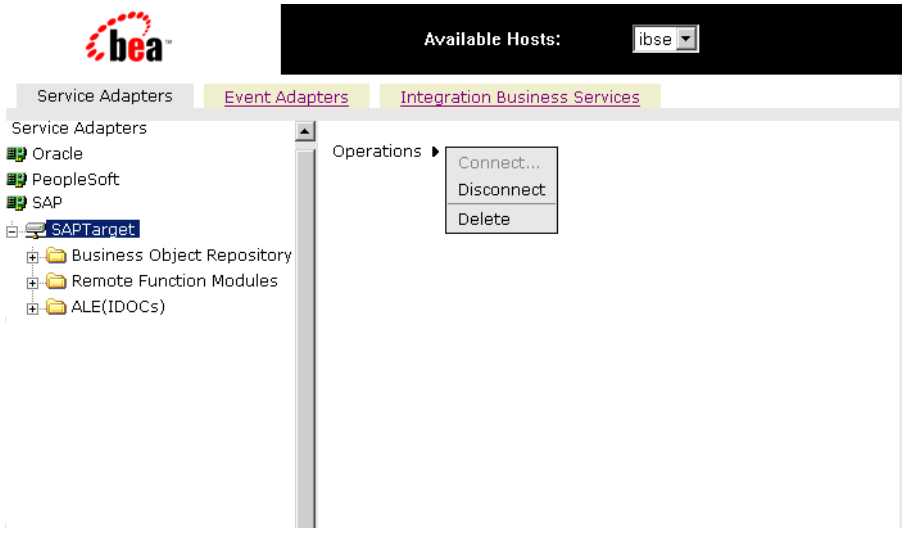
Although you can maintain multiple open connections to different application systems, it is a good practice to close connections when you are not using them.

Procedure: How to Disconnect From a Target

To disconnect from a target:

1. From the left pane, click the target, for example, SAPTarget, to which you are connected.
2. In the right pane, move the pointer over *Operations*.

   The following image shows the target selected in the left pane. In the right pane, the Operations menu appears expanded to display options.



3. Select *Disconnect*.

   Disconnecting from the application system drops the connection, but the node remains.

   In the left pane, the SAPTarget node changes to reflect that a connection was closed (a red 'x' appears) as shown in the following image.

Modifying a Target

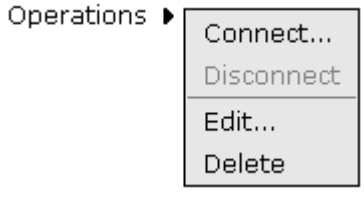
After you create a target for SAP using Servlet Application Explorer, you can edit the information that you provided previously.

Procedure: How to Edit a Target

To edit a target:

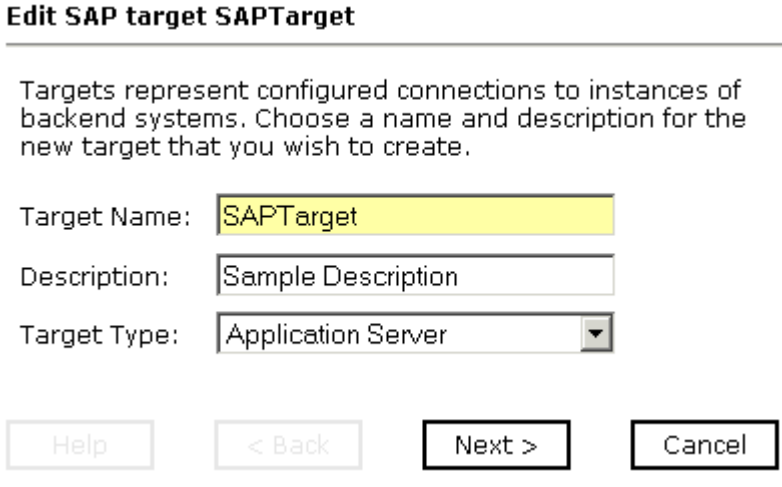
1. In the left pane, click the target, for example, SAPTarget.

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



2. Move the pointer over *Operations* and select *Edit*.

   The Edit pane opens on the right with the target name, a description and a target type selected from the drop-down list as shown in the following image.



3. Modify the connection information.
4. To continue modifying additional information, click *Next*.
5. When you are finished making all of your edits, click *Finish*.

Deleting a Target

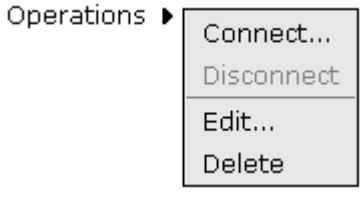
In addition to closing a target, you can delete a target that is no longer required. You can delete it whether or not it is closed. If open, the target automatically closes before it is deleted.

Procedure: How to Delete a Target

To delete a target:

1. In the left pane, click the target, for example, SAPTarget.

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



2. In the right pane, move the pointer over *Operations*.
3. Select *Delete*.

   A confirmation dialog box opens, asking if you want to delete the target.
4. To delete the target you selected, click *OK*.

   The SAPTarget node disappears from the left pane.

Viewing Application System Objects

After you are connected to SAP, Application Explorer enables you to explore and browse business object metadata. For example, Application Explorer enables you to view SAP BAPI, RFC, and iDoc metadata stored in the SAP Business Object repository.

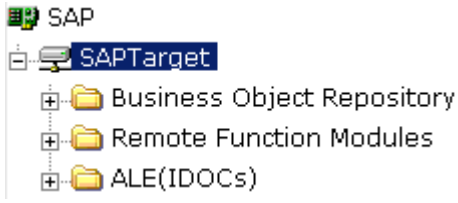
Note: Depending on the release or service pack installed, certain RFCs, for example, RFC_CUSTOMER_GET, may not exist in your particular SAP system. Therefore, the examples included in this documentation may not be relevant to your system. If this is the case, you should use the examples as a general reference for adapter functionality and choose an RFC that exists within your SAP application environment.

Procedure: How to View Application System Objects

To view application system objects:

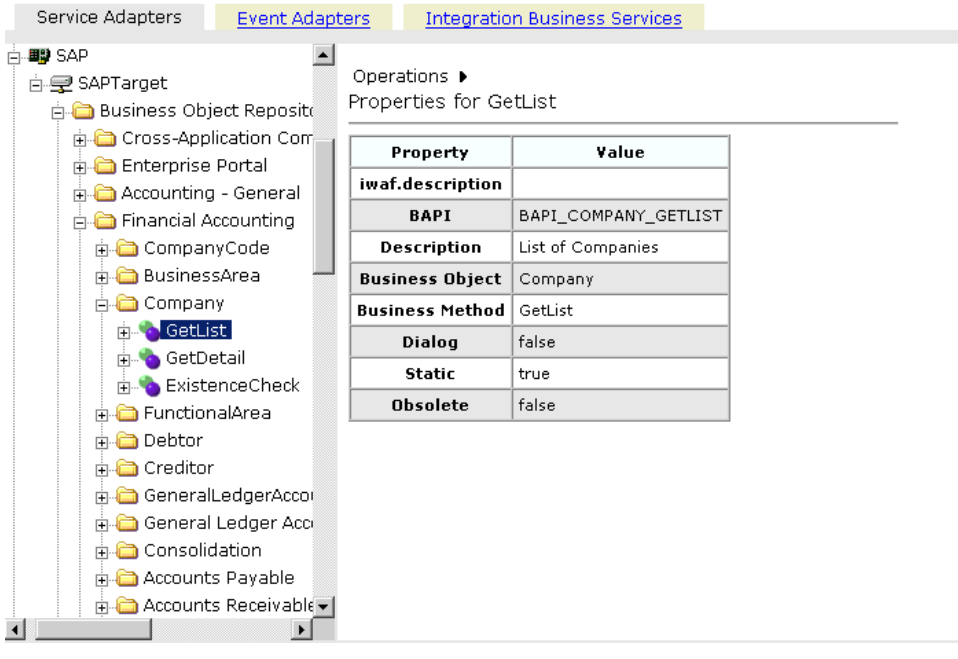
1. Click the icon to the left of the target name, for example, SAPTarget.

This expands the target to expose the available application system objects as shown in the following image.



2. To expand the desired SAP repository node, click the icon to the left of the repository name, for example, Business Object Repository.
3. In the list under Business Object Repository, click the icon next to *Financial Accounting*.

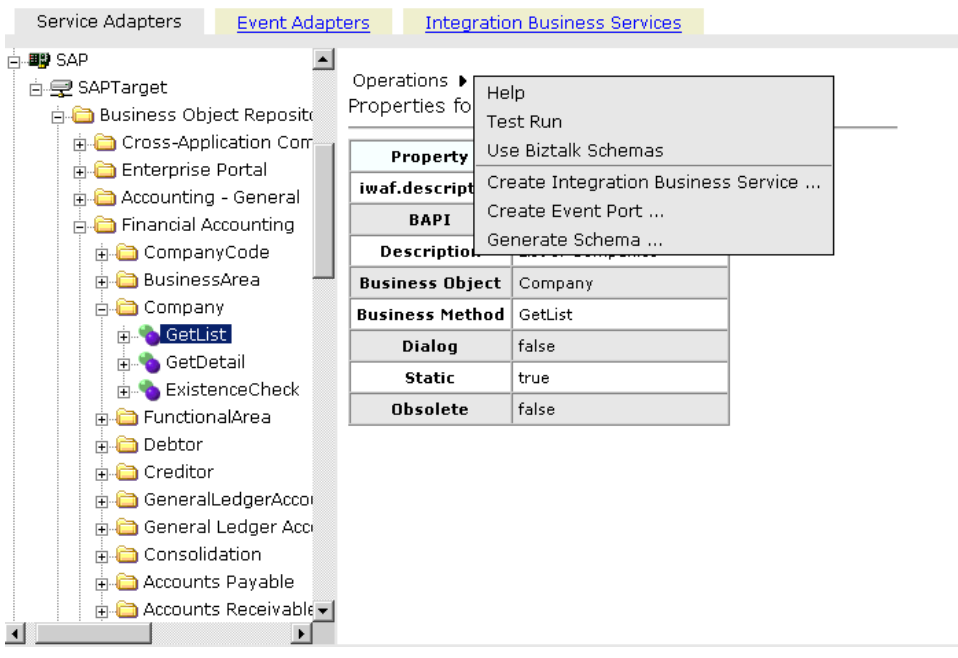
   A list of business objects related to Financial Accounting appears in the left pane. In the right pane, the collapsed Operations menu and a table listing properties and values for the BAPI method named BAPI_COMPANY_GETLIST appears as shown in the following image.



   a. Scroll down and click the icon next to the *Company* business object.

   b. Click the icon next to the BAPI method called *GetList*.
4. In the right pane, move the pointer over *Operations* to view the context menu.

The following image shows the Operations menu expanded over the table listing properties and values for the BAPI method named BAPI_COMPANY_GETLIST in the right pane.



The following options are available from the context menu:

- **Help** provides information about BAPI, RFC, and IDoc usage.
- **Test Run** simulates running the selected RFC or BAPI with sample data you provide.
- **Use Biztalk Schemas** sets the default for schema output that is used by Microsoft Biztalk.
- **Create Integration Business Services** creates Web services for the SAP business object you selected.
- **Create Event Port** creates a port to be used for SAP event handling.
- **Generate Schema** generates XML request and response schemas for the SAP business object you selected.

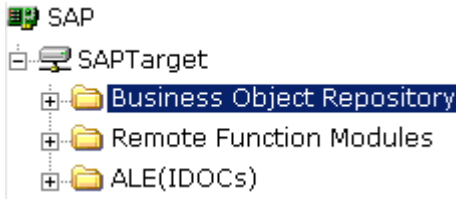
Procedure: How to Search for a Specific SAP Business Object

You can use the search function in Application Explorer to locate an SAP business object.

1. Start Application Explorer and connect to your SAP system target.

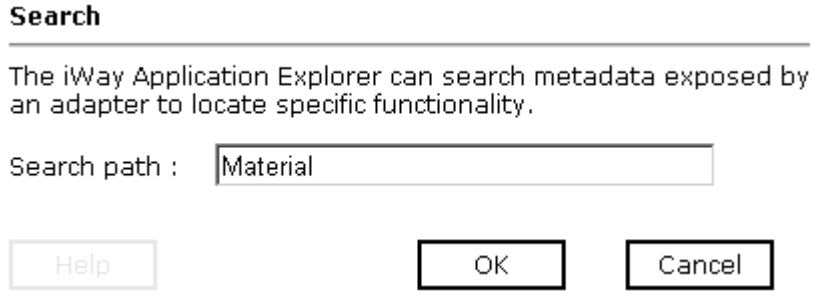
2. Expand the target and select Business Object Repository, Remote Function Modules, or ALE(IDOCs).

   The following image shows Business Object Repository selected in the left pane.



3. In the right pane, move the cursor over *Operations* and select *Search* from the menu.

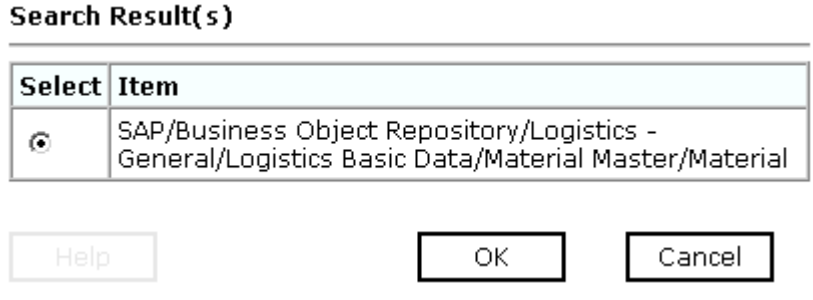
   The following image shows the Search feature that appears in the right pane. It has a search path input area.



4. Enter the name of the business object you want to search for in the Search path text entry box, for example, Material.

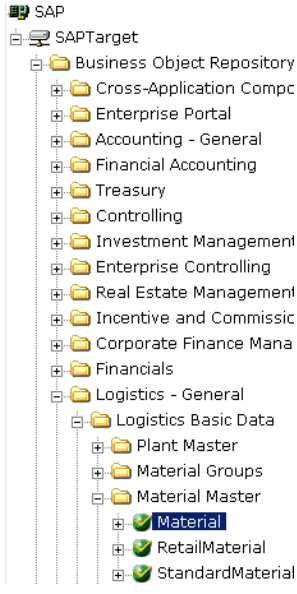
5. Click *OK*.

   The following image show the search results that appear when a search for Material is conducted against the Business Object Repository.



6. Select the radio button next to the item in which you are interested and click *OK*.

Application Explorer locates the business object you selected in the left pane, for example, Material.



Creating an XML Schema

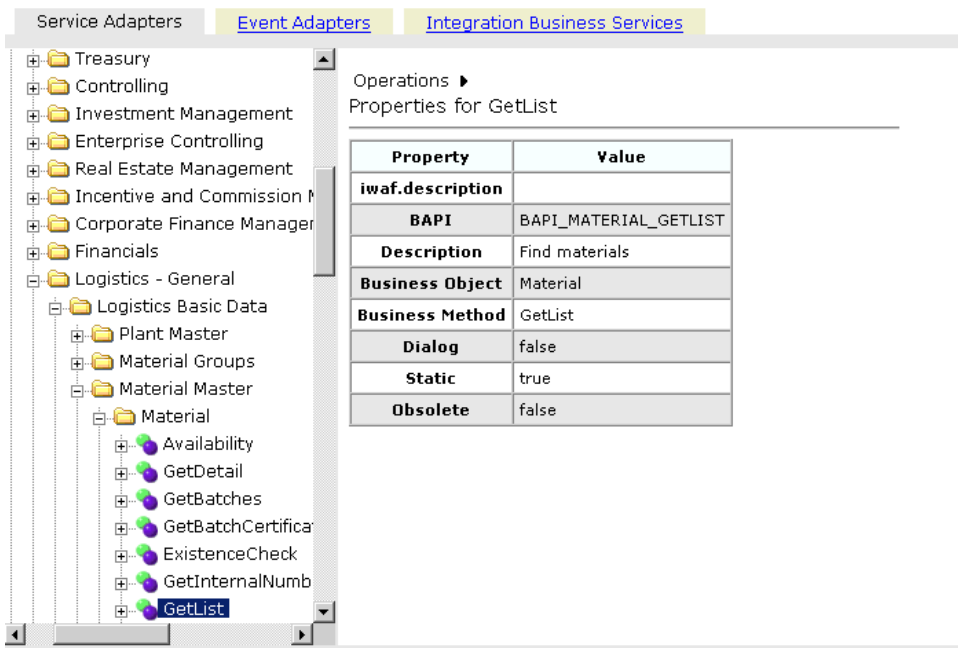
After you browse the SAP business object repository, you can generate XML request and response schemas for the object you wish to use with your adapter.

Procedure: How to Create XML Schemas

To create XML request and response schemas for the SAP BAPI method called BAPI_MATERIAL_GETLIST:

1. In the Business Object Repository, select the *GetList* method.

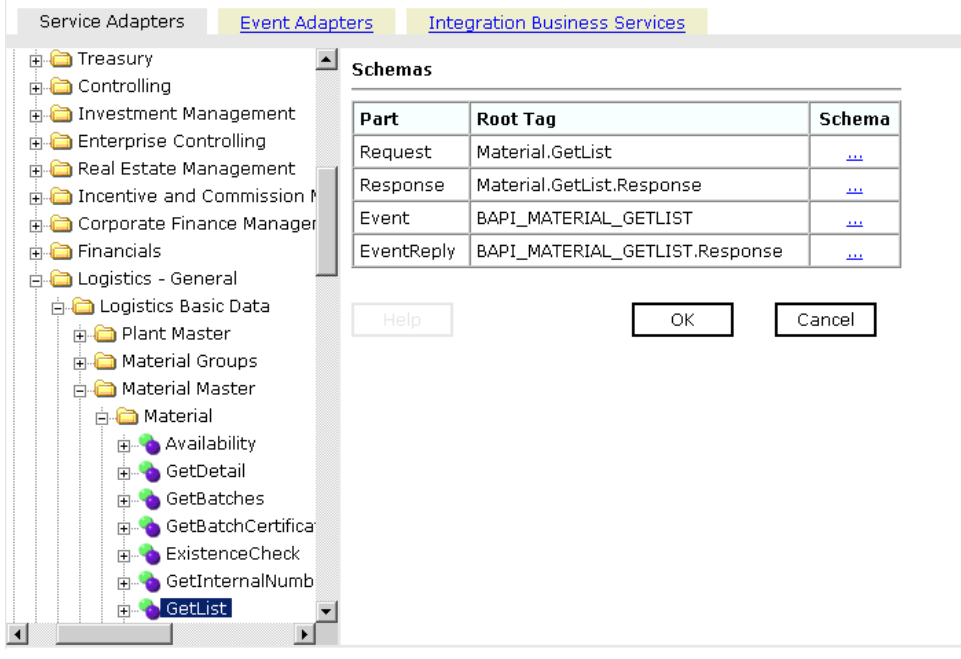
In the right pane, the collapsed Operations menu and a table listing properties and values for the BAPI method named BAPI_MATERIAL_GETLIST appears as shown in the following image.



2. In the right pane, move the pointer over *Operations* and select *Generate Schema* from the menu.

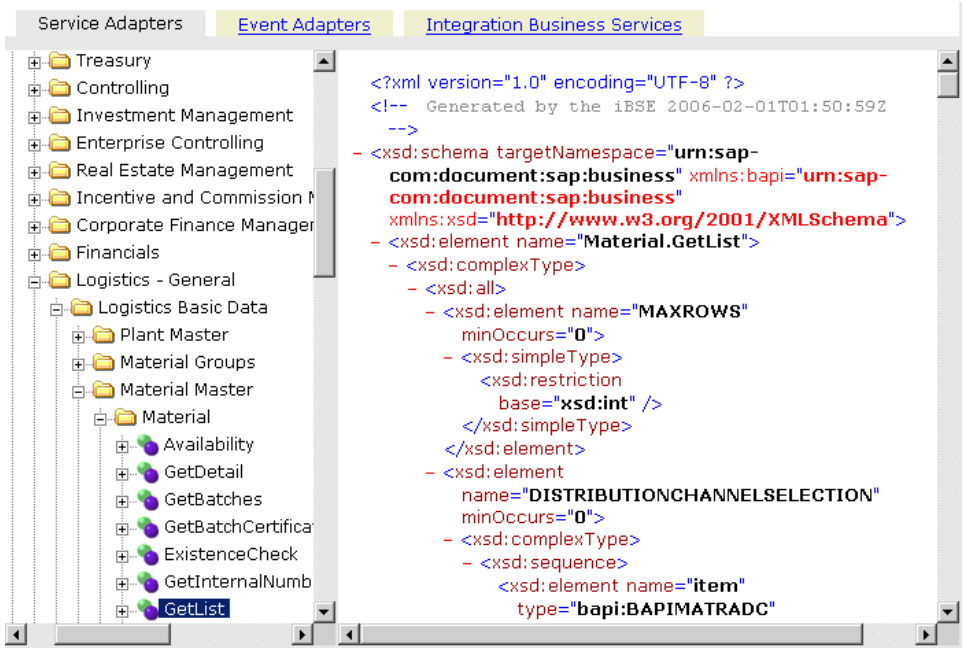
Request, response, and event schemas are created for your business object.

The following image shows the Schemas pane that opens on the right with a table that lists and defines the root tag for each schema and provides hyperlinks to click to view each schema.



3. Click the hyperlink associated with the type of schema you want to view.

For example, if you click the hyperlink for the Request schema, the schema appears in the right pane as shown in the following image.



4. To return to the previous window, click the *Back* button on your Web browser.

After you browse the list of business objects on your SAP system, you can create Integration Business Services. For more information, see Chapter 4, *Creating and Publishing Integration Business Services*.

After the schemas are created, you also can create events. For more information, see Chapter 5, *Configuring the Event Adapter for SAP*.

CHAPTER 4

Creating and Publishing Integration Business Services

Topics:

- Understanding Integration Business Services
- Creating Integration Business Services

This section describes how to create and publish Integration Business Services using Application Explorer.

The functionality of Application Explorer is standard for any deployment type. This section uses the Java™ servlet implementation of Application Explorer to provide examples.

Understanding Integration Business Services

Servlet Application Explorer provides Web developers with a simple, consistent mechanism for extending the capabilities of the BEA WebLogic Adapter for SAP. The Integration Business Services Engine (iBSE) exposes functionality as Web services. It serves as a gateway to heterogeneous back-end applications and databases.

A Web service is a self-contained, modularized function that you can publish and access 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.

After you browse the SAP business object repository and create an XML schema for the object, you can generate an Integration Business Service for the object you wish to use with your adapter.

Creating Integration Business Services

The following topics describe how to create Integration Business Services, including how to generate WSDL (Web Services Description Language) from a Web service.

Creating Business Services With Application Explorer

The following procedure describes how to create Integration Business Services using Servlet Application Explorer. The procedure uses the SAP BAPI method called BAPI_MATERIAL_GETLIST as an example and returns a list of materials.

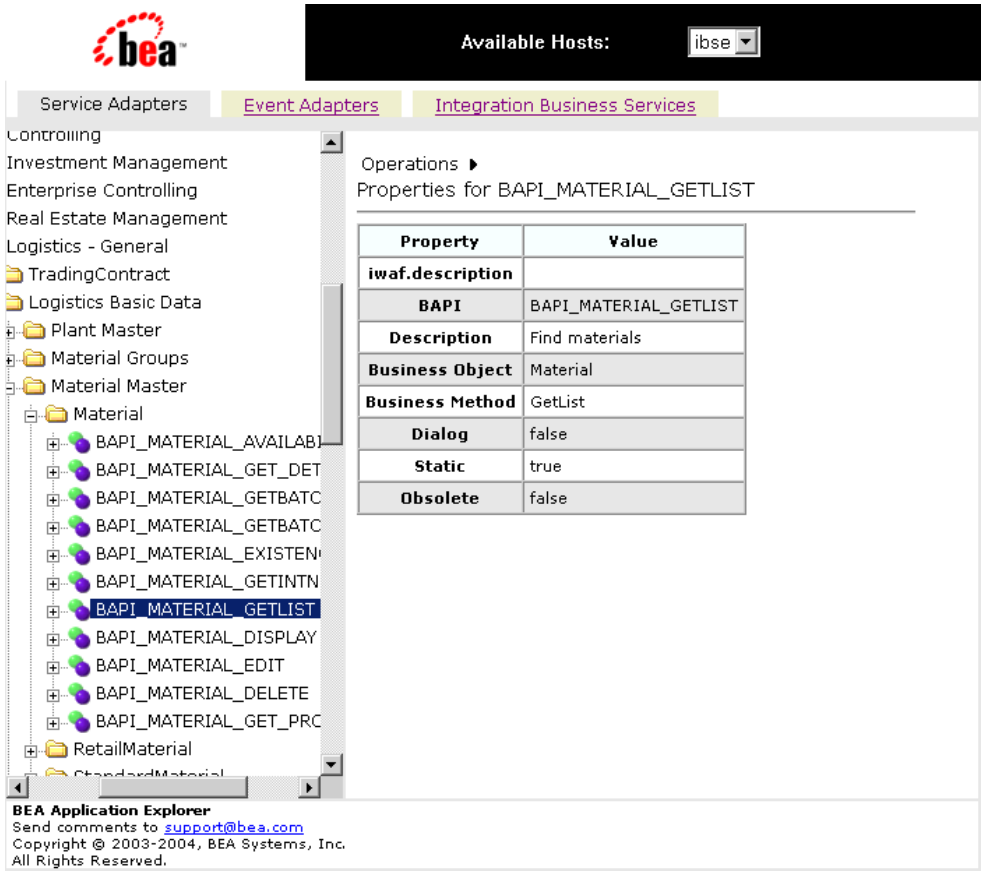
Note: If you want your Web service to use connection pooling, you must specify connection pooling information when connecting or reconnecting to your SAP target.

Procedure: How to Create Integration Business Services

To create Integration Business Services:

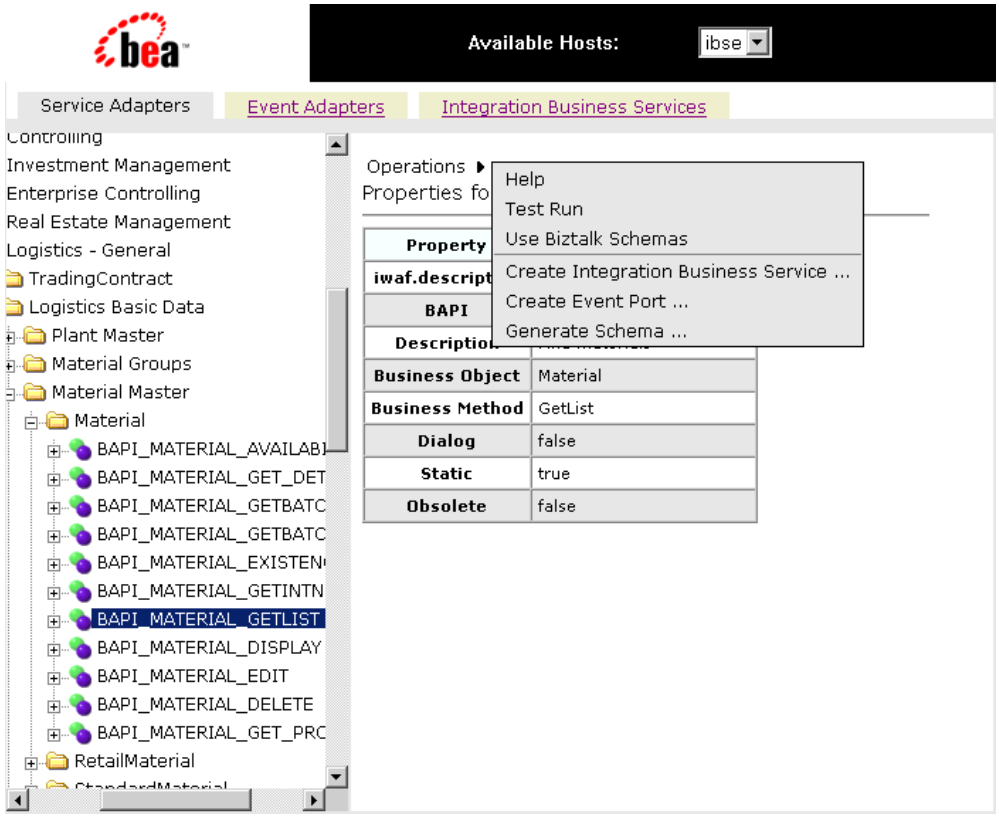
1. From the Business Object Repository, select the *BAPI_MATERIAL_GETLIST* method.

The following image shows the Operations menu in the right pane and a table listing properties and values for the BAPI method called BAPI_MATERIAL_GETLIST.



2. In the right pane, move the pointer over *Operations*.

The Operations menu expands as shown in the following image to display options.

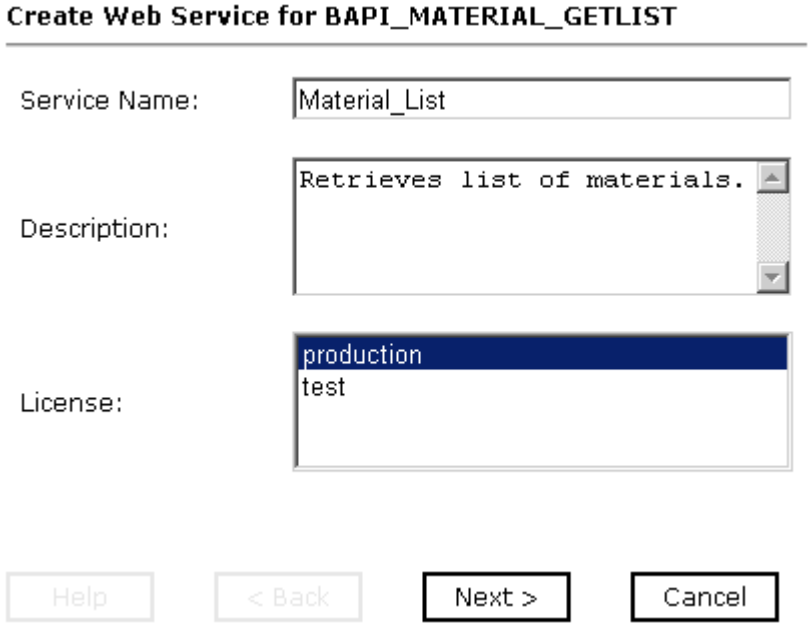


3. Select *Create Integration Business Services*.

The Create Web Service for BAPI_MATERIAL_GETLIST pane opens on the right with options to create a new service or use an existing service.

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

A Create Web Service pane opens where you provide the specific information for the Integration Business Service you are defining as shown in the following image.



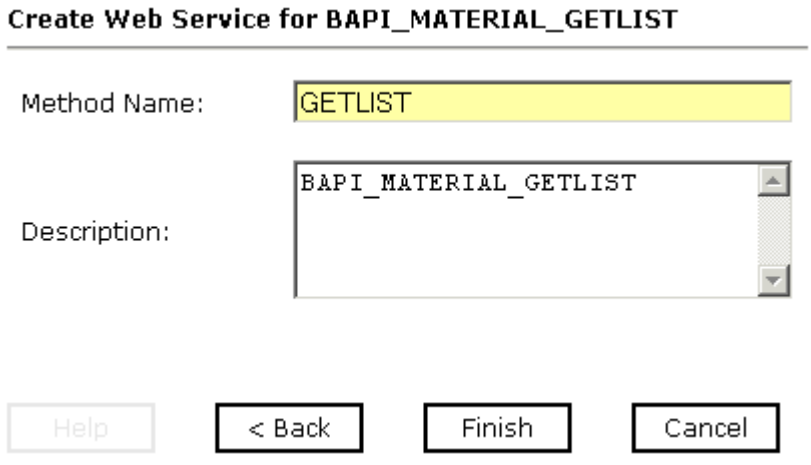
a. In the Service Name field, type a name for the Integration Business Service.

b. In the Description field, type a brief description (optional).

c. In the License field, select the license definition you want to use.

5. Click *Next*.

A second Create Web Service pane opens on the right that includes fields for the method name and a description as shown in the following image.

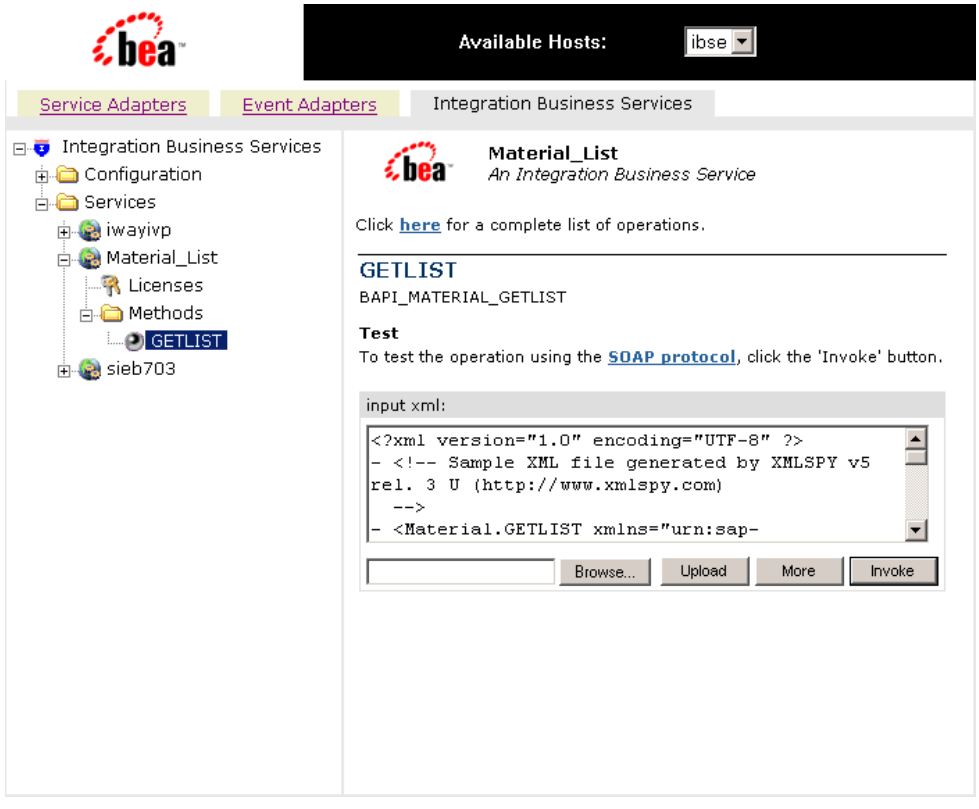


a. In the Method Name field, type a descriptive name for the method.

b. In the Description field, type a brief description for the method (optional).

6. Click *Finish*.

The Integration Business Services tab is active on the right as shown in the following image.



All of the available services that were created appear in the left pane. The Material_List service node is expanded, and the GETLIST method is automatically selected.

The test pane for the GETLIST method opens in the right pane.

7. Enter an XML instance of the schema you generated previously for the SAP business component.

   The document queries the service in the input xml field.

   To use the identical sample input XML illustrated in this example, see *Sample Integration Business Services Input XML* on page 4-9.

8. Click *Invoke*.

The result appears in the right pane as shown in the following image.

Example: Sample Integration Business Services Input XML

The following input XML retrieves a list of materials using the SAP BAPI_MATERIAL_GETLIST method.

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <!-- Sample XML file generated by XMLSPY v5 rel. 3 U
(http://www.xmlspy.com)
  -->
- <Material.GETLIST xmlns="urn:sap-com:document:sap:business"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:sap-com:document:sap:business
C:\temp\service_BAPI_MATERIAL_GETLIST.xsd">
  <MAXROWS>1000</MAXROWS>
- <DISTRIBUTIONCHANNELSELECTION>
- <item>
  <SIGN />
  <OPTION />
  <DISTR_CHAN_LOW />
  <DISTR_CHAN_HIGH />
  </item>
  </DISTRIBUTIONCHANNELSELECTION>
- <MANUFACTURERPARTNUMB>
- <item>
  <MANU_MAT />
  <MFR_NO />
  </item>
  </MANUFACTURERPARTNUMB>
- <MATERIALSHORTDESCSEL>
- <item>
  <SIGN />
  <OPTION />
  <DESCR_LOW />
  <DESCR_HIGH />
  </item>
  </MATERIALSHORTDESCSEL>
- <MATNRLIST>
- <item>
  <MATERIAL />
  <MATL_DESC />
  <MATERIAL_EXTERNAL />
  <MATERIAL_GUID />
  <MATERIAL_VERSION />
  </item>
  </MATNRLIST>
- <MATNRSELECTION>
- <item>
  <SIGN>E</SIGN>
```

```
  <OPTION>BT</OPTION>
  <MATNR_LOW>1000</MATNR_LOW>
  <MATNR_HIGH>1010</MATNR_HIGH>
  </item>
  </MATNRSELECTION>
- <PLANTSELECTION>
- <item>
  <SIGN />
  <OPTION />
  <PLANT_LOW />
  <PLANT_HIGH />
  </item>
  </PLANTSELECTION>
- <RETURN>
- <item>
  <TYPE />
  <ID />
  <NUMBER />
  <MESSAGE />
  <LOG_NO />
  <LOG_MSG_NO />
  <MESSAGE_V1 />
  <MESSAGE_V2 />
  <MESSAGE_V3 />
  <MESSAGE_V4 />
  <PARAMETER />
  <ROW>0</ROW>
  <FIELD />
  <SYSTEM />
  </item>
  </RETURN>
- <SALESORGANISATIONSELECTION>
- <item>
  <SIGN />
  <OPTION />
  <SALESORG_LOW />
  <SALESORG_HIGH />
  </item>
  </SALESORGANISATIONSELECTION>
- <STORAGELOCATIONSELECT>
- <item>
  <SIGN />
  <OPTION />
  <STLOC_LOW />
  <STLOC_HIGH />
  </item>
  </STORAGELOCATIONSELECT>
  </Material.GETLIST>
```

Generating WSDL From a Web Service

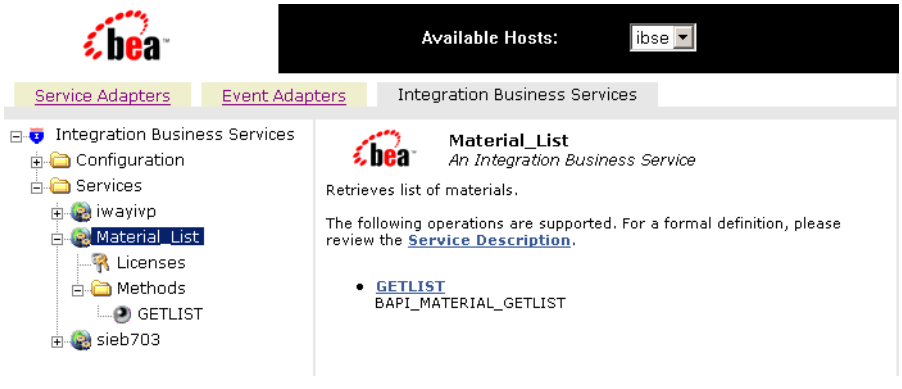
Generating WSDL (Web Services Description Language) from a Web service enables you to make the Web service available to other services within a host server such as the BEA WebLogic Server.

Procedure: How to Generate WSDL From a Web Service

To generate WSDL from a Web service:

1. Click the *Integration Business Services* tab.
2. To view the service for which you want to generate WSDL, in the left pane, expand the list of services.
3. Select the service, for example, Material_List.

   The following image shows Material_List service selected in the left pane. The link for the service appears in the right pane.



   a. Right-click the *Service Description* link.

   b. Select *Save Target As*.

4. Choose a location for the file and add a .wsdl file extension.
5. Click *Save*.

   For example, saving a Web service called Material_List for an SAP R/3 creates a file named Material_List.wsdl.

   Note: The file extension must be .wsdl.

The following is an example of a WSDL file for a Web service called BAPI_MATERIAL_GET_DETAIL.

```
<definitions xmlns:rfc="urn:iwaysoftware:ibse:jul2003:BAPI:response"
 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:BAPI: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:BAPI"
 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:schema>
  <xs:schema targetNamespace="urn:schemas-iwaysoftware-com:iwse"
   elementFormDefault="qualified">
    <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 xmlns:rfc="urn:iwaysoftware:ibse:jul2003:BAPI"
   targetNamespace="urn:iwaysoftware:ibse:jul2003:BAPI"
   xmlns:m1="urn:iwaysoftware:ibse:jul2003:BAPI"
   elementFormDefault="qualified">
     <xs:element name="BAPI">
       <xs:complexType>
```

```
        <xs:sequence>
          <xs:element name="BAPI_MATERIAL_GET_DETAIL">
            <xs:complexType>
              <xs:all>
                <xs:element minOccurs="1" name="MATERIAL">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <xs:maxLength value="18"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:element>
                <xs:element minOccurs="0" name="PLANT">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <xs:maxLength value="4"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:element>
                <xs:\element minOccurs="0" name="VALUATIONAREA">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <xs:maxLength value="4"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:element>
                <xs:element minOccurs="0" name="VALUATIONTYPE">
                  <xs:simpleType>
                    <xs:restriction base="xs:string">
                      <xs:maxLength value="10"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:element>
              </xs:all>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
  <xs:schema xmlns:rfc="urn:iwaysoftware:ibse:jul2003:BAPI:response"
   targetNamespace="urn:iwaysoftware:ibse:jul2003:BAPI:response"
   xmlns:m11="urn:iwaysoftware:ibse:jul2003:BAPI:response"
   elementFormDefault="qualified">
    <xs:element name="BAPIResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="BAPI_MATERIAL_GET_DETAIL.Response">
            <xs:complexType>
```

```
                <xs:all>
                   <xs:element type="rfc:BAPIMATDOC" minOccurs="0"
                    name="MATERIALPLANTDATA"/>
                   <xs:element type="rfc:BAPIMATDOBEW" minOccurs="0"
                    name="MATERIALVALUATIONDATA"/>
                   <xs:element type="rfc:BAPIMATDOA" minOccurs="0"
                    name="MATERIAL_GENERAL_DATA"/>
                   <xs:element type="rfc:BAPIRETURN" minOccurs="0"
                    name="RETURN"/>
                </xs:all>
             </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attribute type="xs:string" use="required" name="cid"/>
      </xs:complexType>
    </xs:element>
    <xs:complexType name="BAPIMATDOC">
      <xs:sequence>
        <xs:element name="PUR_GROUP">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="3"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="ISSUE_UNIT">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="3"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="BAPIMATDOBEW">
      <xs:sequence>
        <xs:element name="PRICE_CTRL">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="1"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="MOVING_PR">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="23"/>
            </xs:restriction>
```

```
        </xs:simpleType>
      </xs:element>
      <xs:element name="STD_PRICE">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="23"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="PRICE_UNIT">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="5"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="CURRENCY">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="5"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="CURRENCY_ISO">
        <xs:simpleType>
           <xs:restriction base="xs:string">
             <xs:maxLength value="3"/>
           </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="BAPIMATDOA">
    <xs:sequence>
      <xs:element name="MATL_DESC">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="40"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="OLD_MAT_NO">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="18"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
```

```
<xs:element name="MATL_TYPE">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="4"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="IND_SECTOR">
   <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="1"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="DIVISION">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="2"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="MATL_GROUP">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="9"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="PROD_HIER">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="18"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="BASIC_MATL">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="14"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="STD_DESCR">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="18"/>
    </xs:restriction>
  </xs:simpleType>
```

```
</xs:element>
<xs:element name="LAB_DESIGN">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="3"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="PROD_MEMO">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="18"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="PAGEFORMAT">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="4"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="CONTAINER">
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:maxLength value="2"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="STOR_CONDS">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="2"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="TEMP_CONDS">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="2"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="BASE_UOM">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="3"/>
    </xs:restriction>
```

```
      </xs:simpleType>
    </xs:element>
    <xs:element name="EAN_UPC">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:maxLength value="18"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="EAN_CAT">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:maxLength value="2"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="SIZE_DIM">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="32"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="GROSS_WT">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="13"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="NET_WEIGHT">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="13"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="UNIT_OF_WT">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="3"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="VOLUME">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="13"/>
```

```
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="VOLUMEUNIT">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="3"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="LENGTH">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="13"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="WIDTH">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="13"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="HEIGHT">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="13"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="UNIT_DIM">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="3"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="MANU_MAT">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:maxLength value="40"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="MFR_NO">
   <xs:simpleType>
     <xs:restriction base="xs:string">
```

```
            <xs:maxLength value="10"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="BASE_UOM_ISO">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="3"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="UNIT_OF_WT_ISO">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="3"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="VOLUMEUNIT_ISO">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="3"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="UNIT_DIM_ISO">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="3"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="CREATED_ON">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="8"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="CREATED_BY">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:maxLength value="12"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="LAST_CHNGE">
        <xs:simpleType>
```

```
            <xs:restriction base="xs:string">
              <xs:maxLength value="8"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="CHANGED_BY">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="12"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="MATL_CAT">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="2"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="EMPTIESBOM">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="1"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="BASIC_MATL_NEW">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="48"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="BAPIRETURN">
      <xs:sequence>
        <xs:element name="TYPE">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="1"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="CODE">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:maxLength value="5"/>
```

```
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="MESSAGE">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="220"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="LOG_NO">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="20"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="LOG_MSG_NO">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="6"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="MESSAGE_V1">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="50"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="MESSAGE_V2">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="50"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="MESSAGE_V3">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="50"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="MESSAGE_V4">
    <xs:simpleType>
      <xs:restriction base="xs:string">
```

```
          <xs:maxLength value="50"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:schema>
</types>
<message name="BAPIIn"><part element="m1:BAPI" name="parameters"/>
</message>
<message name="BAPIOut"><part element="m11:BAPIResponse"
 name="parameters"/>
</message>
<message name="BAPI_MATERIAL_GET_DETAILHeader">
 <part element="tns:ibsinfo" name="header"/>
</message>
<message name="AdapterException"><part element="tns:adapterexception"
 name="fault"/>
</message>
<portType name="BAPI_MATERIAL_GET_DETAILSoap">
 <operation name="BAPI">
  <documentation/>
   <input message="tns:BAPIIn"/>
   <output message="tns:BAPIOut"/>
   <fault message="tns:AdapterException" name="AdapterExceptionFault"/>
  </operation>
</portType>
<binding type="tns:BAPI_MATERIAL_GET_DETAILSoap"
 name="BAPI_MATERIAL_GET_DETAILSoap">
  <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="BAPI">
    <soap:operation style="document"
     soapAction="BAPI_MATERIAL_GET_DETAIL.BAPIRequest@test@@"/>
    <input>
      <soap:body use="literal"/>
      <soap:header part="header"
       message="tns:BAPI_MATERIAL_GET_DETAILHeader" use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
    <fault name="AdapterExceptionFault">
      <soap:fault use="literal" name="AdapterExceptionFault"/>
    </fault>
  </operation>
</binding>
<service name="BAPI_MATERIAL_GET_DETAIL">
```

```
    <documentation>BAPI_MATERIAL_GET_DETAIL
    </documentation>
    <port binding="tns:BAPI_MATERIAL_GET_DETAILSoap"
     name="BAPI_MATERIAL_GET_DETAILSoap1">
      <soap:address
       location="http://GERBER-2K.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 user name and password values that you specify in the SOAP header must be valid. The values are used to connect to SAP. The user name and password values that you provided for SAP when you created a target 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>
```

CHAPTER 5

Configuring the Event Adapter for SAP

Topics:

- Understanding Event Functionality
- Creating, Editing, or Deleting a Port
- Creating, Editing, or Deleting a Channel
- Deploying Components in a Clustered BEA WebLogic Environment
- Synchronous Event Processing

This section describes how to use Application Explorer to connect to SAP and listen for events.

The functionality of Application Explorer is standard for any deployment type. This section uses the Java™ servlet implementation of Application Explorer to provide examples.

Understanding Event Functionality

Events are generated as a result of activity on an application system. You can use events to trigger an action in your application.

Applications or functions within SAP may broadcast processing information at predefined points, called events. You must configure an event listener if you are to receive events from SAP. For example, the SAP Business Object, Material, may raise the event status Material, assigned when a material is created. If you wish to consume this event, you must configure an event listener to capture this event within SAP and transmit the event notification to your system.

To create an event, you must create a port and a channel using Application Explorer.

The following is a description of how ports and channels work.

- 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 more information, see *Creating, Editing, or Deleting a Port on page 5-2*.

- 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, Editing, or Deleting a Channel* on page 5-16.

Creating, Editing, or Deleting a Port

The following procedures describe how to create, edit, or delete an event port using Servlet Application Explorer. You can create a port for an SAP business function from the Service Adapters tab or from the Event Adapters tab.

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

- File
- iBSE
- MSMQ
- JMSQ
- SOAP
- WLWS

- HTTP
- MQSeries

The following dispositions are available when using Application Explorer in conjunction with a JCA connector implementation:

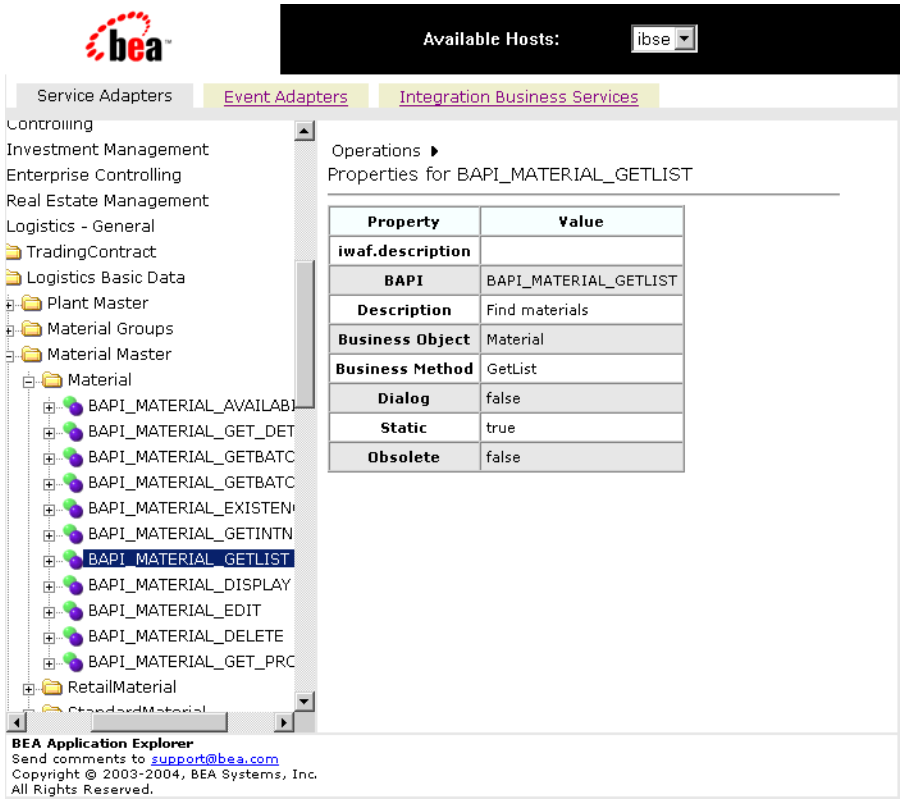
- File
- JMS
- HTTP

Procedure: How to Create a Port for the File Disposition

To create a port for the File disposition using Application Explorer:

1. Click the *Service Adapters* tab.
2. From the Business Object Repository, select the *BAPI_MATERIAL_GETLIST* method.

The following image shows the Operations menu in the right pane and a table listing properties and values for the BAPI method named BAPI_MATERIAL_GETLIST.



3. In the right pane, move the pointer over *Operations* and select *Create Event Port*.

The Create Event Port pane opens on the right as shown in the following image.

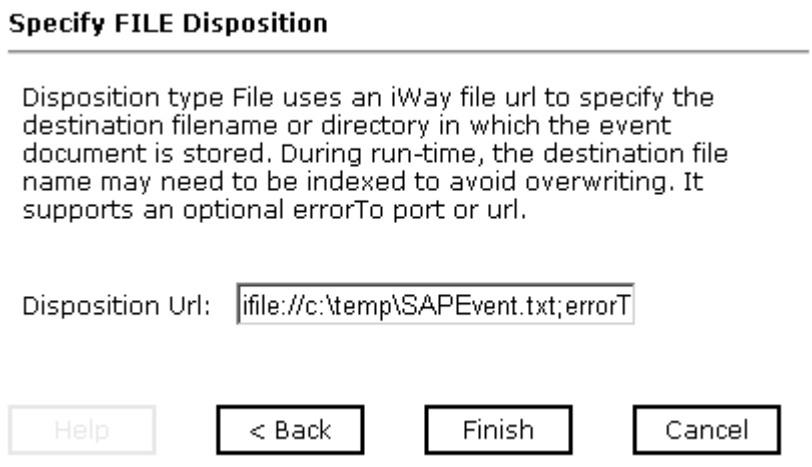
a. In the Event Port Name field, type a name.

b. In the Event Port Description field, type a brief description (optional).

c. From the Disposition Protocol drop-down list, select *FILE*.

4. Click *Next*.

The Specify FILE Disposition pane opens on the right as shown in the following image and includes information about the File disposition.



5. In the Disposition Url field, type a destination where 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.

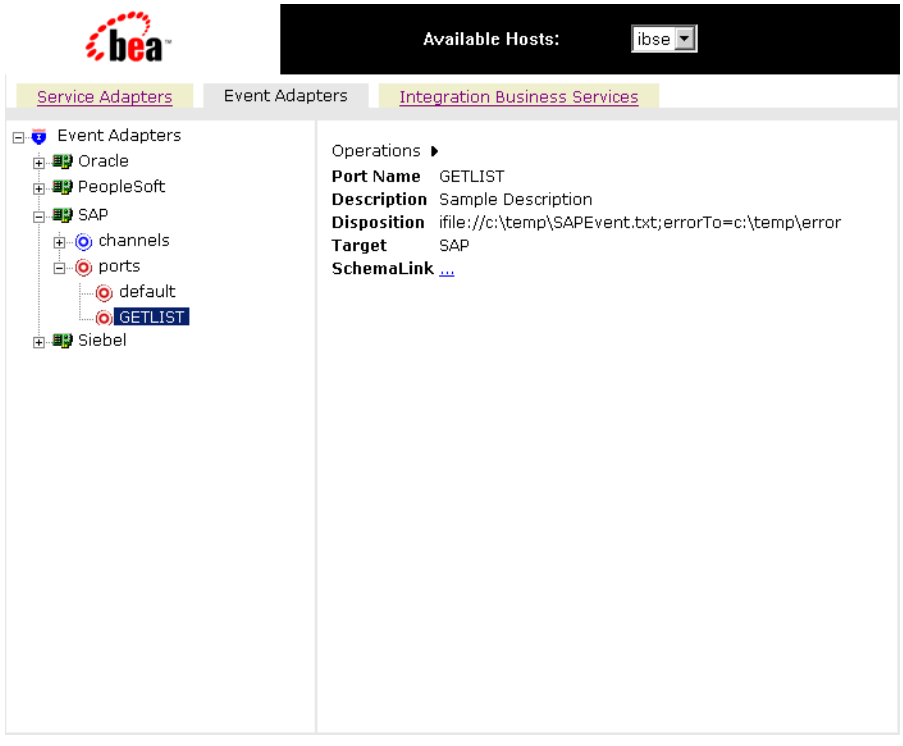
Parameter	Description
location	Full directory path and file name to which the data is written.
errorDest	Location to which error logs are sent. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.

For example:

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

6. Click *Finish*.

The Event Adapters tab becomes available. The event port appears under the ports node in the left pane. In the right pane, a summary of the information associated with the port appears.



7. To view the event schema that was created for the event port, click *SchemaLink*.

You are now ready to associate the event port for File with a channel. For more information, see *Creating, Editing, or Deleting a Channel* on page 5-16.

Procedure: How to Create a Port for the iBSE Disposition

The iBSE disposition enables an event to launch an Integration Business Services method. To create a port for an iBSE disposition using Application Explorer:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *SAP* node.
3. Select the *ports* node.
4. Move the pointer over *Operations* and select *Add a new port*.

   The Create New Port pane opens on the right.

a. In the Name field, type a name.

b. In the Description field, type a brief description (optional).

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=responseTo];[errorTo=errorDest]
```

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

Parameter	Description
svcName	Name of the service created with iBSE.
mthName	Name of the method created for the Web service.
responseTo	Location to which responses to the Web service are posted. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.
errorDest	Location to which error logs are sent. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.

5. Click *OK*.

The port appears under the ports node in the left pane. In the right pane, a summary of the information associated with the port appears.

Procedure: How to Create a Port for the MSMQ Disposition

The MSMQ disposition supports public and private queues. To create a port for an MSMQ disposition using Application Explorer:

1. Click the *Event Adapters* tab.

2. In the left pane, expand the *SAP* node.

3. Select the *ports* node.

4. Move the pointer over *Operations* and select *Add a new port*.

The Create New Port pane opens on the right.

a. In the Name field, type a name.

b. In the Description field, type a brief description (optional).

c. From the Disposition Protocol drop-down list, select *MSMQ*.

d. In the Disposition field, enter an MSMQ destination in the format:

`msmq://host/private$/qName;[errorTo=errorDest]`

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

Parameter	Description
host	Name of the host on which the Microsoft Queuing system runs.
queueType	Type of queue. For private queues, enter *Private$*. 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.
qName	Name of the queue in which messages are placed.
errorTo	Location where error documents are sent. Predefined port name or another full URL. Optional.

5. Click *OK*.

The port appears under the ports node in the left pane. In the right pane, a summary of the information associated with the port appears.

Procedure: How to Create a Port for the JMS Queue Disposition

The JMS queue disposition enables an event to be enqueued to a JMS queue. To create a port for a JMS queue disposition using Application Explorer:

1. Click the *Event Adapters* tab.

2. In the left pane, expand the *SAP* node.

3. Select the *ports* node.

4. Move the pointer over *Operations* and select *Add a new port*.

The Create New Port window opens in the right pane.

a. In the Name field, type a name.

b. In the Description field, type a brief description (optional).

c. From 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, use the following format:

```
zjmsq:myQueueName@myQueueFac;jndiurl=[myurl];
jndifactory=[myfactory];user=[user];password=[xxx];
errorTo=[pre-defined port name or another disposition url]
```

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

```
jms:jmsqueue@jmsfactory;jndiurl=;jndifactory=;
```

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

Parameter	Description
queue	JNDI name of a queue to which events are emitted.
Connection Factory	A resource that contains information about the JMS Server. The WebLogic connection factory is: `javax.jms.QueueConnectionFactory`

jndiurl	The URL to use to contact the JNDI provider. The syntax of this URL depends on which JNDI provider is being used. This value corresponds to the standard JNDI property,
	`java.naming.provider.url`
	For BEA WebLogic Server this is:
	`t3://host:port`
	where:
	`host`
	Is the machine name where WebLogic Server is installed.
	`port`
	Is the port on which WebLogic server is listening. The default port if not changed at installation is 7001.

jndifactory	Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider. For WebLogic Server, the WebLogic factory is: `weblogic.jndi.WLInitialContextFactory`

Parameter	Description
user	Valid user name required to access a JMS server.
password	Valid password required to access a JMS server.
errorTo	Location where error documents are sent. A predefined port name or another full URL. Optional.

5. Click *OK*.

The port appears under the ports node in the left pane. In the right pane, a summary of the information associated with the port appears.

Procedure: How to Create a Port for the SOAP Disposition

To create a port for a SOAP disposition using Application Explorer:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *SAP* node.
3. Select the *ports* node.
4. Move the pointer over *Operations* and select *Add a new port*.

   The Create New Port pane opens on the right.

   a. In the Name field, type a name.

   b. In the Description field, type a brief description (optional).

   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 lists and describes the parameters for the disposition.

Parameter	Description
wsdl-url	The URL to the WSDL file that is required to create the SOAP message, for example: `http://localhost:7001/ibse/IBSEServlet/test/`*`webservice`*`.ibs?wsdl` where: *`webservice`* Is the name of the Web service you created using Application Explorer. To find this value, you can navigate to the Integration Business Services tab and open the Service Description link in a new window. The WSDL URL appears in the Address field. Alternatively, you can open the WSDL file in a third-party XML editor (for example, XMLSPY) and view the SOAP request settings.
soapaction	Method that is called by the SOAP disposition. This value can be found in the WSDL file.
method	Web service method you are using. Value is found in the WSDL file.
namespace	XML namespace you are using. Value is found in the WSDL file.
responseTo	Location to which responses are posted. Can be a predefined port name or another URL. Optional. The URL must be complete, including the protocol.
errorTo	Location where error documents are sent. A predefined port name or another full URL. Optional.

Note: To use the SOAP disposition with a synchronous event, use Remote Function Modules to generate the schema and WSDL file instead of using the Business Object Repository for any RFC or BAPI.

5. Click *OK*.

The event port appears under the ports node in the left pane. In the right pane, a summary of the information associated with the port appears.

Procedure: How to Create a Port for the HTTP Disposition

The HTTP disposition uses an HTTP URL to specify an HTTP end point to which the event document is posted. To create a port for an HTTP disposition using Application Explorer:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *SAP* node.
3. Select the *ports* node.
4. Move the pointer over *Operations* and select *Add a new port*.

   The Create New Port pane opens on the right.

   a. In the Name field, type a name.

   b. In the Description field, type a brief description (optional).

   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
url	The URL target for the post operation.
respDest	Location to which responses are posted. A predefined port name or another full URL. Optional. 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.
uri	Universal resource identifier that completes the URL specification.

5. Click *OK*.

The event port appears under the ports node in the left pane. In the right pane, a summary of the information associated with the port appears.

Procedure: How to Create a Port for the MQSeries Disposition

The MQSeries disposition enables an event to be enqueued to an MQSeries queue. Both queue manager and queue name may be specified. To create a port for an MQSeries disposition using Application Explorer:

1. Click the *Event Adapters* tab.
2. In the left pane, expand the *SAP* node.
3. Select the *ports* node.
4. Move the pointer over *Operations* and select *Add a new port*.

   The Create New Port pane opens on the right.

   a. In the Name field, type a name.

   b. In the Description field, type a brief description (optional).

   c. From 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=[port];channel=[channnelname];
      errorTo=[pre-defined port name or another disposition url]
      ```

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

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

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

Parameter	Description
qManager	Name of the queue manager to which the server must connect.
qName or respqueue	Name of the queue where messages are placed.
host	Host on which the MQ Server is located (MQ Client only).

Parameter	Description
port	Number to connect to an MQ Server queue manager (MQ client only).
channel	Case-sensitive name of the channel that connects with the remote MQ Server queue manager (MQ client only). The default channel name for MQSeries is SYSTEM.DEF.SVRCONN.
errorTo	Location where error documents are sent. This can be a predefined port name or another full URL. Optional.

5. Click *OK*.

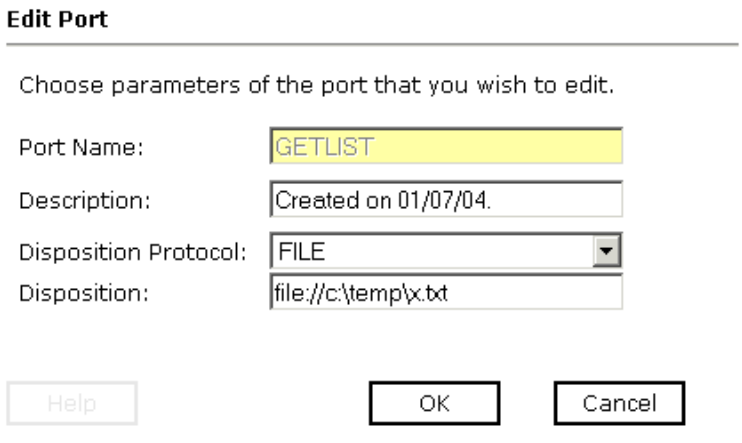
The port appears under the ports node in the left pane. In the right pane, a summary of the information associated with the port appears.

Procedure: How to Edit an Event Port

To edit an event port:

1. Select the event port you want to edit.
2. In the right pane, move the pointer over *Operations* and select *Edit*.

   The Edit Port pane opens on the right, with fields where you can modify the description, disposition protocol, and disposition of the port as shown in the following image. You cannot change the port name.



3. Make the required changes to the event port configuration fields and click *OK*.

Procedure: How to Delete an Event Port

To delete an event port:

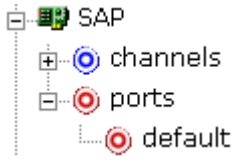
1. Select the event port you want to delete.
2. In the right pane, move the pointer over *Operations* and select *Delete*.

   A confirmation dialog box opens, asking whether to delete the event port.
3. To delete the event port you selected, click *OK*.

The event port disappears from the list in the left pane.

Using the Default Event Port

When using Application Explorer to connect to SAP and listen for events, a default event port is available at all times as shown in the following image.



The default event port can be used for testing purposes or when you do not want to route event data to a specific port you configured. The default port is enabled when you start a channel that does not have a specific event port assigned.

The default event data is actually a file disposition that writes to an out.xml file in the following output directory:

```
ifile://./eventOut/out.xml
```

If you have deployed your adapter in a BEA WebLogic environment, this directory is created in your user domain.

Creating, Editing, or Deleting a Channel

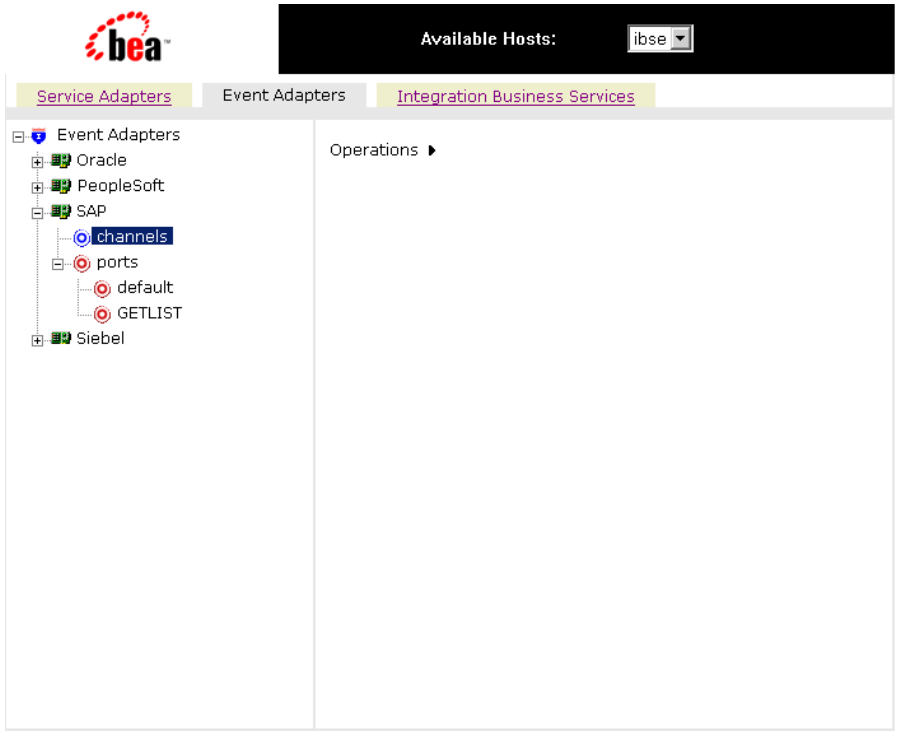
The following procedures describe how to create, edit, or delete a channel for your event adapter as well as how to start or stop a channel. All defined event ports must be associated with a channel. You can create a channel using Servlet Application Explorer.

Procedure: How to Create a Channel

To create a channel using Application Explorer:

1. Click the *Event Adapters* tab.

The list of adapters that support events appears in the left pane as shown in the following image.

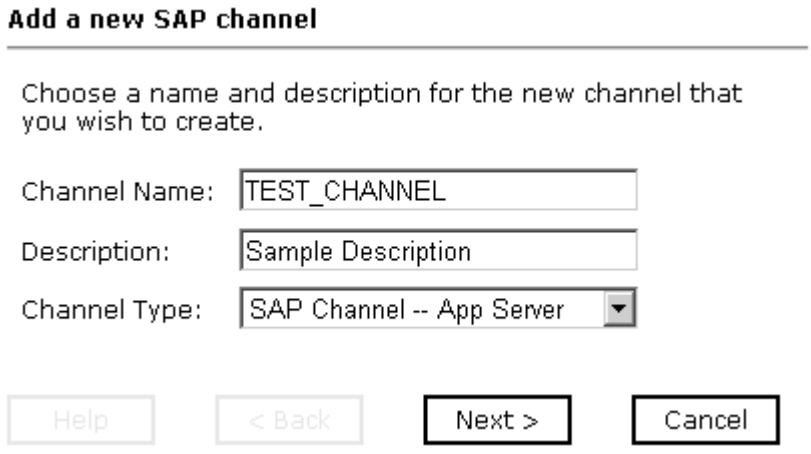


2. Expand the *Event Adapters* node, for example, SAP.

   The ports and channels nodes appear in the left pane.

3. Click the *channels* node.
4. In the right pane, move the pointer over *Operations* and select *Add a new channel*.

The Add a new SAP channel pane opens on the right as shown in the following image.



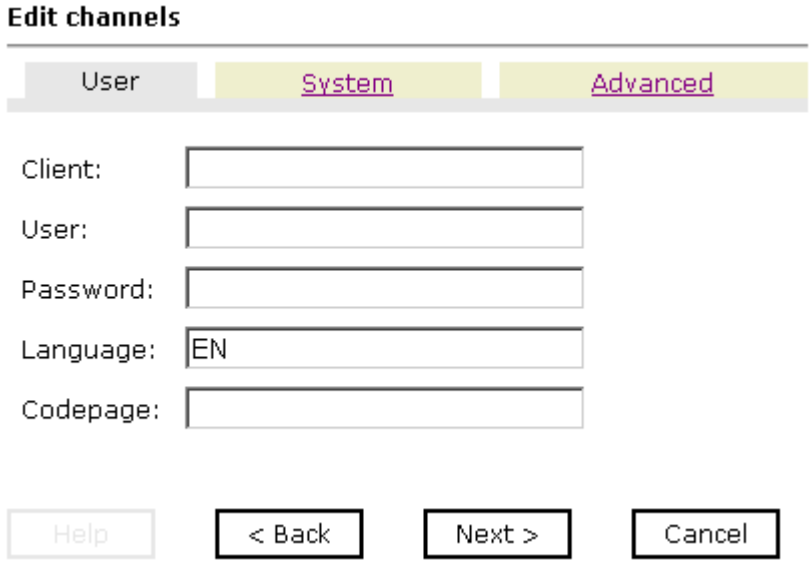
a. In the Channel Name field, type a name, for example, TEST_CHANNEL.

b. In the Description field, type a brief description (optional).

c. From the Channel Type drop-down list, select *SAP Channel -- App Server*.

5. Click *Next*.

The Edit channels pane opens on the right as shown in the following image, with fields for the client, user, password, language, and code page information for the channel. The User tab is active.



a. On the User tab, enter the information that is specific to your SAP system.

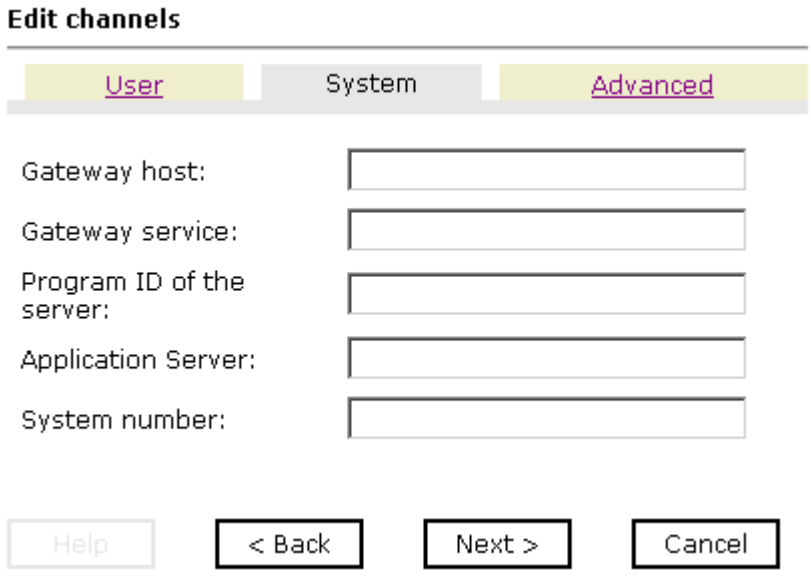
The following table lists and describes the user parameters.

Parameter	Description
Client	Client number defined for the SAP system for client communications.
User	Valid user ID for the SAP system.
Password	Valid password for the SAP system.
Language	Valid language key. EN (English) is the default value.
Codepage	Character code page value used by the SAP system (optional).

Note: For more information on obtaining the values found in the User tab, see your SAP system administrator.

b. Click the *System* tab.

The System tab becomes active as shown in the following image, with fields to enter a gateway host, gateway service, program ID of the server, application server, and system number for the channel.



c. Enter the user information that is specific to your SAP system.

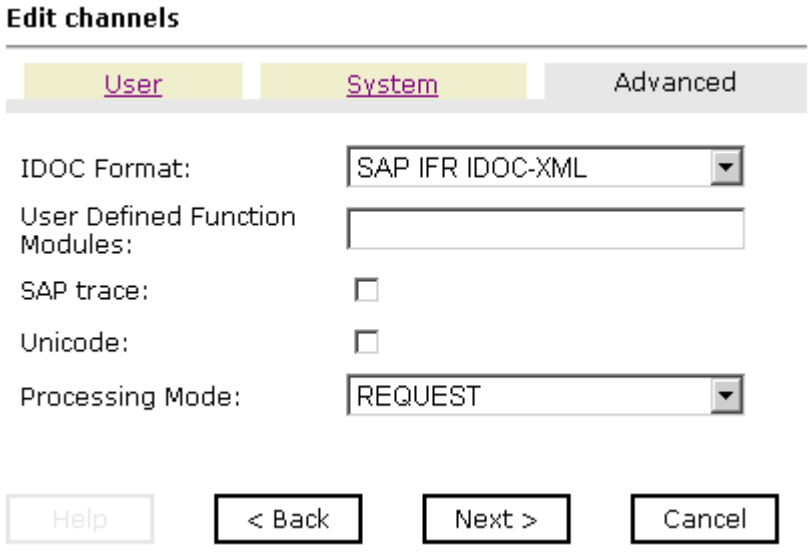
The following table lists and describes the system parameters.

Parameter	Description
Gateway host	Host name of the SAP gateway.
Gateway service	Service of the SAP gateway.
Program ID of the server	Program ID of the registered server program. **Note:** The program ID of the server is case sensitive.
Application Server	Host name of the SAP system.
System number	Number of the SAP system (two-digit numeric value).

Note: For more information on obtaining the values found in the System tab, see your SAP system administrator.

d. Click the *Advanced* tab.

The Advanced tab becomes active as shown in the following image, with fields to enter or select an IDoc format, user defined function modules, and processing mode options, as well as check boxes to enable SAP traces or Unicode encoding of the event data.



e. Specify additional information or criteria for the channel you are creating.

The following table lists and describes the parameters on the Advanced tab.

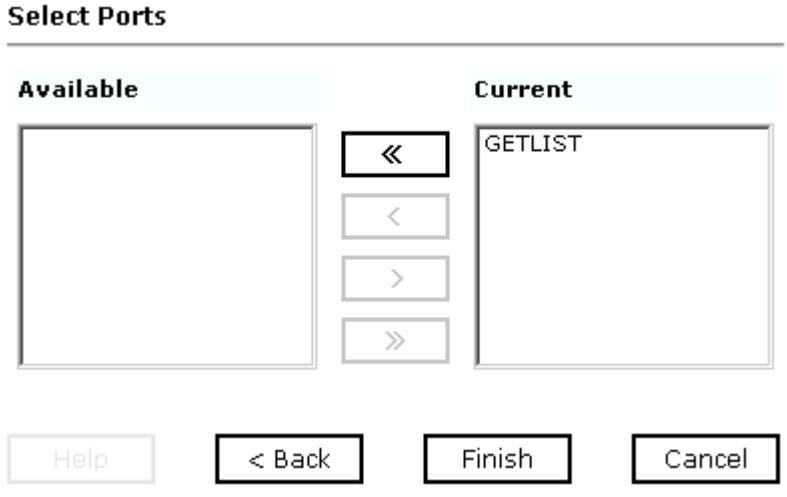
Parameter	Description
IDOC Format	The IDoc format to use for the channel you are creating. Possible values include: SAP IFR IDOC-XML (Default) FLAT IDOC
User Defined Function Modules	Path on the file system that points to your user-defined function module. A user-defined function module is used as a data source.
SAP trace	Select this option to enable SAP traces.
Unicode	Select this option if you want your response to have a Unicode format.

Parameter	Description
Processing Mode	Processing mode to use for your channel. Possible values include: REQUEST (Default) REQUEST_RESPONSE

Note: For more information on obtaining the values found in the Advanced tab, see your SAP system administrator.

6. Click *Next*.

The Select Ports pane opens as shown in the following image where you can move ports between the Current list and the Available list.

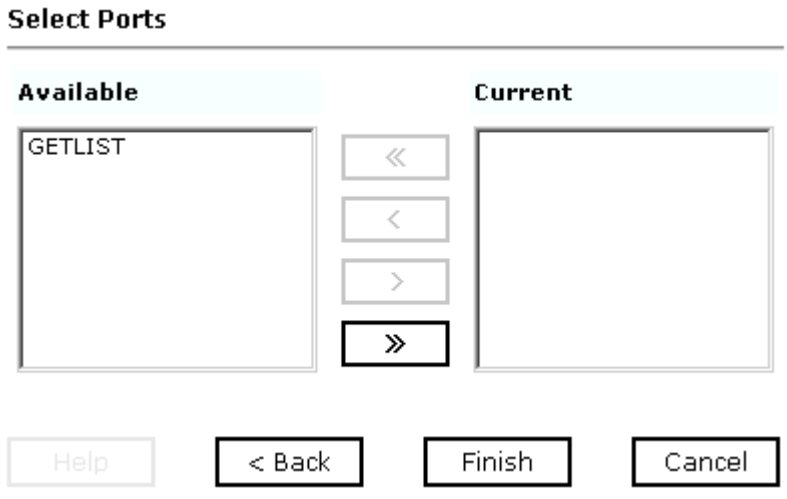


a. Select an event port from the list of current ports.

b. To transfer the port to the list of available ports, click the single left arrow button or to associate all event ports, click the double left arrow button.

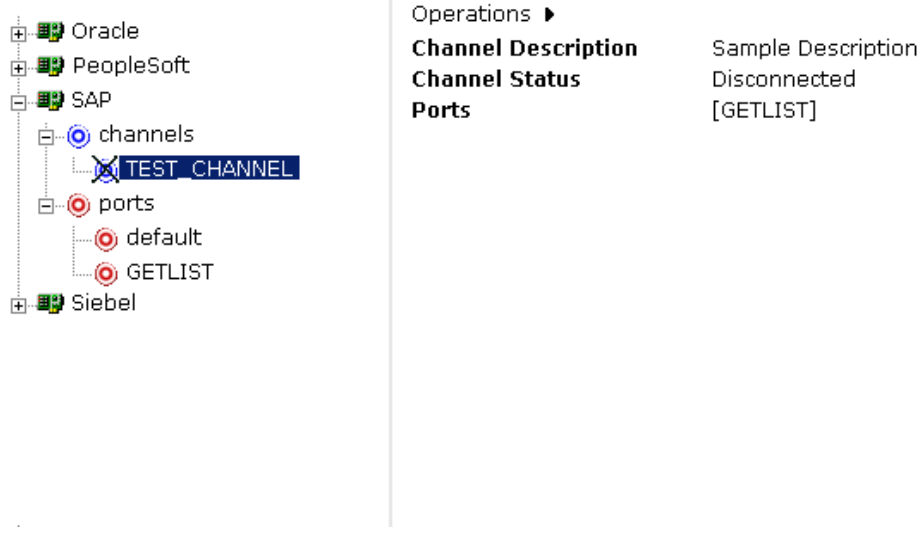
When only one port appears, it is as if you are transfering all ports. Therefore, only the double arrow button is active.

The port appears in the list of available ports as shown in the following image.

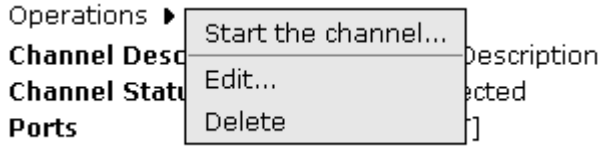


7. Click *Finish*.

In the left pane, the TEST_CHANNEL is selected and has an 'X' over the icon. The summary pane opens on the right with the channel description, channel status, and available ports as shown in the following image.

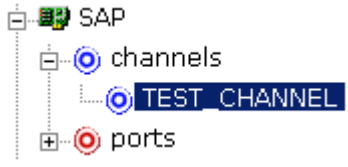


All the information is associated with the channel you created. The 'X' over the icon indicates that the channel is currently disconnected. To activate your event configuration, you must start the channel using the Operations menu, as shown in the following image.



8. Move the pointer over *Operations* and select *Start the channel*.

   The channel you created is now active, and the 'X' disappears, as shown in the following image.



   Note: You can start a channel even if a specific event port is not assigned. A default port is used in these situations. However, it is recommended that you always assign a specific port to the channel you create.

   For more information on creating a port, see *Creating, Editing, or Deleting a Port on page 5-2*.

   For more information on using the default port, see *Using the Default Event Port* on page 5-16.

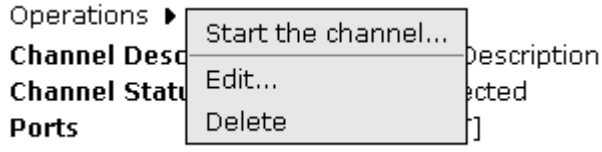
9. To stop the channel at any time, move the pointer over *Operations* and select *Stop the channel*.

Procedure: How to Edit a Channel

To edit an existing channel:

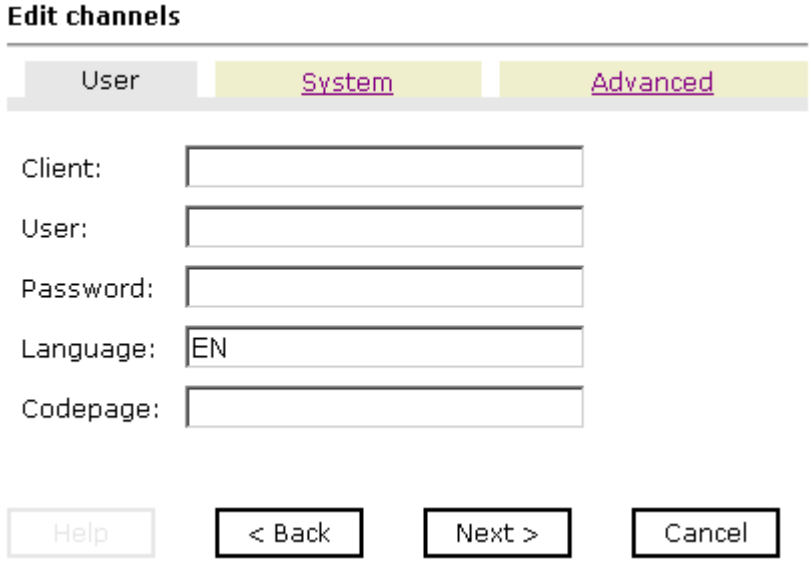
1. In the left pane, select the channel you want to edit.

The Operations menu opens in the right pane over the summary of the channel information, as shown in the following image.



2. Move the pointer over *Operations* and select *Edit*.

   The Edit channels pane opens on the right as shown in the following image, with the User tab active.



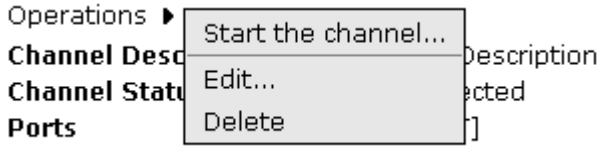
3. Make the required changes to the channel configuration fields.
4. To continue editing information on the other tabs, click *Next*.
5. When you complete your edits, click *Finish*.

Procedure: How to Delete a Channel

To delete an existing channel:

1. In the left pane, select the channel you want to delete.

   The Operations menu opens in the right pane over the summary of the channel information, as shown in the following image.



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

   A confirmation dialog box opens asking if you want to delete this item.

3. To delete the channel you selected, click *OK*.

   The channel disappears from the list in the left pane.

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. For more information, see the BEA clustering documentation.
- 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, configure an administrative server to manage the managed servers.
2. Add and configure as many managed servers as required.
3. Add and configure an HTTP router. This does not have to be a part of WebLogic and can be an outside component.
4. If you configure the HTTP router within 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.
5. Add the managed servers to your cluster/clusters.

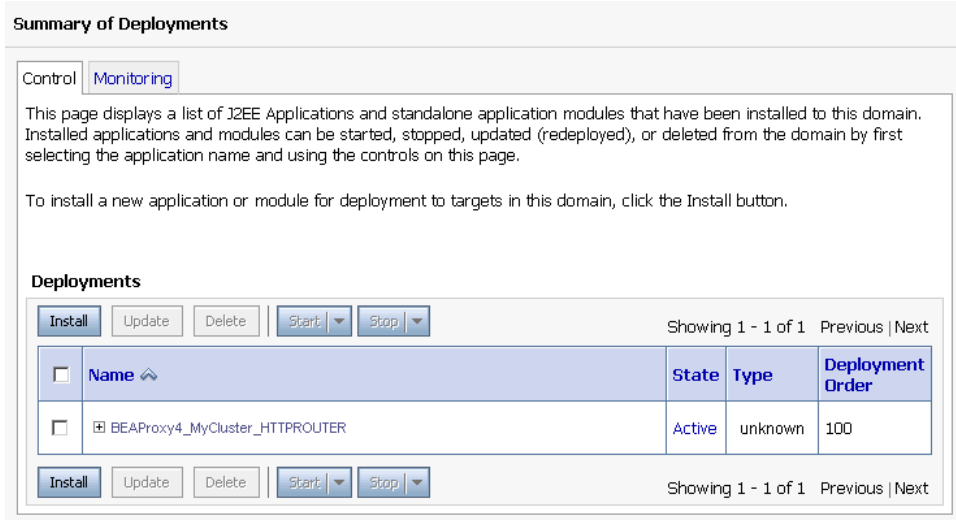
   For more information on configuring the BEA WebLogic Server in a clustered environment, see *Deploying WebLogic Integration Solutions*.

Procedure: How to Deploy iBSE in a Clustered Environment

To deploy iBSE in a clustered environment:

1. Start the BEA WebLogic Server and open the WebLogic Server Console.
2. In the left pane, click *Lock & Edit* and click *Deployments*.

   The Summary of Deployments page opens. Notice that the cluster you configured using the BEA Configuration Wizard is added to the list of deployments in the WebLogic Server Console.



3. Click *Install*.

   A page appears where you can specify the location of the file or directory you wish to deploy.

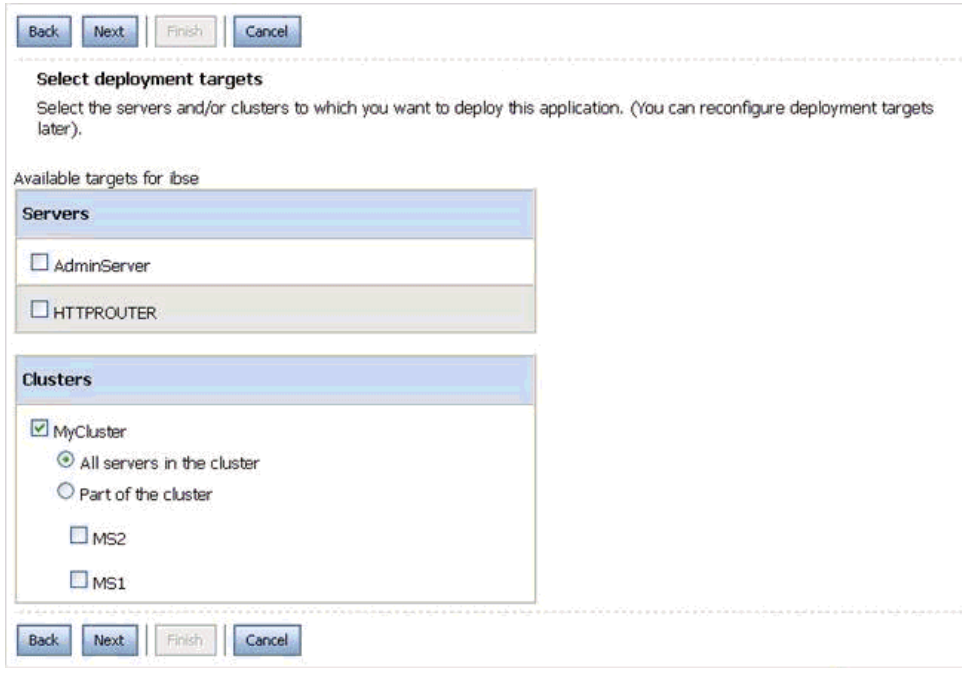
4. Deploy iBSE to the cluster by clicking the links below *Location* and then selecting the radio button next to the ibse directory, for example:

   ```
   C:\Program Files\iWay55\bea\ibse
   ```

5. Click the radio button next to the ibse directory and click *Next*.

6. Click *Next* again, leaving the default *Install this deployment as an application* option selected.

   The Select deployment targets page opens.



7. In the Clusters section, click the checkbox next to the name of the cluster you configured, for example *MyCluster*.
8. Click the *All servers in the cluster* radio button and click *Next*.
9. Click *Next* again, leaving the default values.
10. Click *Finish* to complete the deployment.
11. On the left, click *Activate Changes*.
12. On the right, click *Control*.
13. Click the *ibse* checkbox.
14. Click *Start* and select *Servicing All Requests* from the drop-down menu.
15. Click *Yes*.

    The State of the iBSE application is now *Active*.

    Note: iBSE must use a database repository (SQL Server, Oracle, DB2, or Sybase). Do not use a file repository. You can select this in the Repository Type drop-down list in the iBSE monitoring page. For more information, see the *BEA WebLogic ERP Adapter Installation and Configuration* documentation. 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

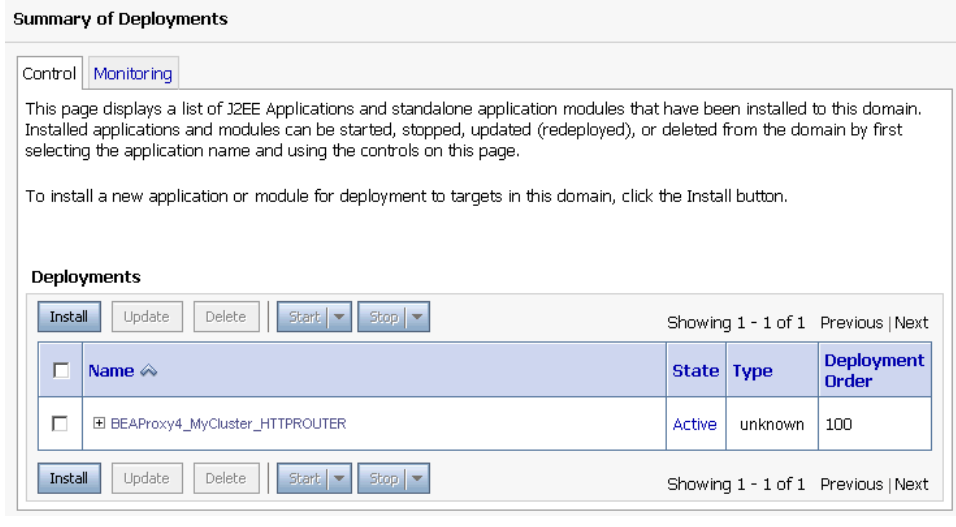
Is the port specific to each server, since you deploy iBSE to an entire cluster. For example, 8001, 8002, or any other port that is specified for each managed node.

Procedure: How to Deploy Servlet Application Explorer to an Admin Server

To deploy Servlet Application Explorer to an Admin server:

1. Start the BEA WebLogic Server and open the WebLogic Server Console.
2. In the left pane, click *Lock & Edit* and click *Deployments*.

   The Summary of Deployments page opens. Notice that the cluster you configured using the BEA Configuration Wizard is added to the list of deployments in the WebLogic Server Console.



3. Click *Install*.

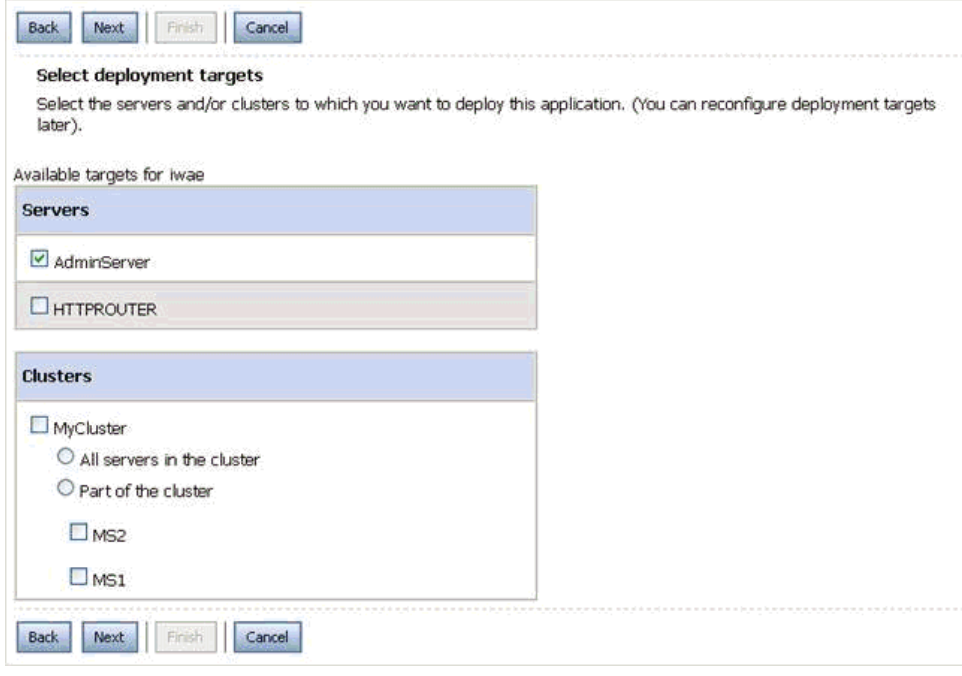
   A page appears where you can specify the location of the file or directory you wish to deploy.

4. Deploy Servlet Application Explorer to the cluster by clicking the links below *Location* and then selecting the radio button next to the ibse directory, for example:

```
C:\Program Files\iWay55\bea\iwae
```

5. Click *Next* after selecting the radio button next to the iwae directory.
6. Click *Next* again, leaving the default *Install this deployment as an application* option selected.

   The Select deployment targets page opens.



7. In the Servers section, click the AdminServer checkbox and click *Next*.
8. Click *Next* again, leaving the default values.
9. Click *Finish* to complete the deployment.
10. On the left, click *Activate Changes*.
11. On the right, click *Control*.
12. Click the *iwae* checkbox.
13. Click *Start* and select *Servicing All Requests* from the drop-down menu.
14. Click *Yes*.

    The State of the iwae application is now *Active*.

Procedure: How to Configure Ports and Channels in a Clustered Environment

You can use Servlet Application Explorer to configure ports and channels in a clustered environment.

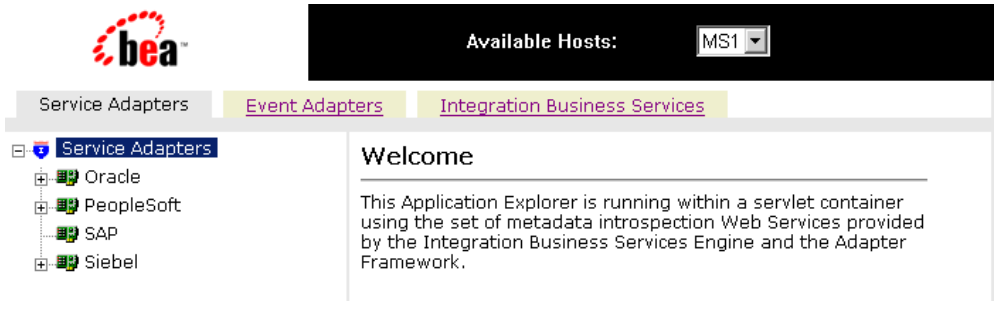
Note: Before using Servlet Application Explorer in a clustered environment, you must edit the web.xml file and specify the correct URL to your iBSE deployment. The default location on Windows is:

```
C:\Program Files\iWay55\bea\iwae\WEB-INF\web.xml
```

For more information on configuring the web.xml file for the Servlet Application Explorer, see the *BEA WebLogic ERP Adapter Installation and Configuration* documentation.

To configure ports and channels in a clustered environment:

1. Start Servlet Application Explorer.



   The Available Hosts drop-down menu in the upper right lists the JCA or iBSE instances in your cluster, for example, MS1 and MS2.

2. Click the *Event Adapters* tab.
3. Select an adapter from the adapter list (in this example, SAP) and add a new port. For more information, see *Creating, Editing, or Deleting a Port on page 5-2*.
4. Create a channel and add the port you created. For more information, see *Creating, Editing, or Deleting a Channel* on page 5-16.
5. Enter the application server parameters for your channel.
6. Start the channel.
7. Select the second iBSE instance, for example MS2, from the Available Hosts drop-down menu.

   The connection to iBSE must be configured to each instance of the managed server.

   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.

Synchronous Event Processing

You can configure synchronous event processing using Application Explorer to trigger a Web service after an event occurs in the SAP system. The event response that is received can then be routed to another disposition for further processing.

Procedure: How to Configure Synchronous Event Processing Using Application Explorer

To configure synchronous event processing:

1. Create a Web service for an SAP Remote Function Module, for example, BAPI_MATERIAL_GETLIST.
2. View the WSDL file.
3. In the Create New Port pane, create a port using the SOAP disposition.
   a. In the Name field, type a name.
   b. In the Description field, type a brief description (optional).
   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 lists and describes the parameters for the disposition.

Parameter	Description
wsdl-url	The URL to the WSDL file that is required to create the SOAP message, for example: `http://localhost:7001/ibse/IBSEServlet/test/`*`webservice`*`.ibs?wsdl` where: *`webservice`* Is the name of the Web service you created using Application Explorer. To find this value, you can navigate to the Integration Business Services tab and open the Service Description link in a new window. The WSDL URL appears in the Address field. Alternatively, you can open the WSDL file in a third-party XML editor (for example, XMLSPY) and view the SOAP request settings.
soapaction	Method that is called by the SOAP disposition.This value can be found in the WSDL file.
method	Web service method you are using. This value is in the WSDL file.
namespace	The XML namespace you are using. This value is in the WSDL file.
responseTo	Location to which responses are posted. Can be a predefined port name or another URL. Optional. The URL must be complete, including the protocol.
errorTo	Location where error documents are sent. This can be a predefined port name or another full URL. Optional.

Note: To use the SOAP disposition with a synchronous event, use Remote Function Modules to generate the schema and WSDL file instead of using the Business Object Repository for any RFC or BAPI.

The following is an example of a completed SOAP disposition:

```
soap:http://localhost:7001/ibse/IBSEServlet/test/soapWS.ibs?wsdl;
soapaction=soapWS.GETLISTRequest@test@@;method=GETLIST;
namespace=urn:iwaysoftware:ibse:jul2005:GETLIST;
responseTo=ifile://c:\output\sap\soapOut.xml
```

4. Using the Edit channels pane, create a channel.
   a. Provide the required information to connect to SAP in the System and User tabs.
   b. In the Advanced tab, from the Processing Mode drop-down list, select *REQUEST_RESPONSE*.
5. Associate the port you created earlier with the new channel.
6. Start the channel.

   A Web service for an SAP Remote Function Module, for example, BAPI_MATERIAL_GETLIST, is triggered after an event occurs in the SAP system. The response document is returned and routed to a file location.

CHAPTER 6

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 Application Explorer, you can use management and monitoring tools to measure the performance in 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 that enables you to manage and monitor services and events currently in use. The console also displays 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://hostname:port/ibse/IBSEConfig`

   where:

   hostname

   Is the machine where the application server is running.

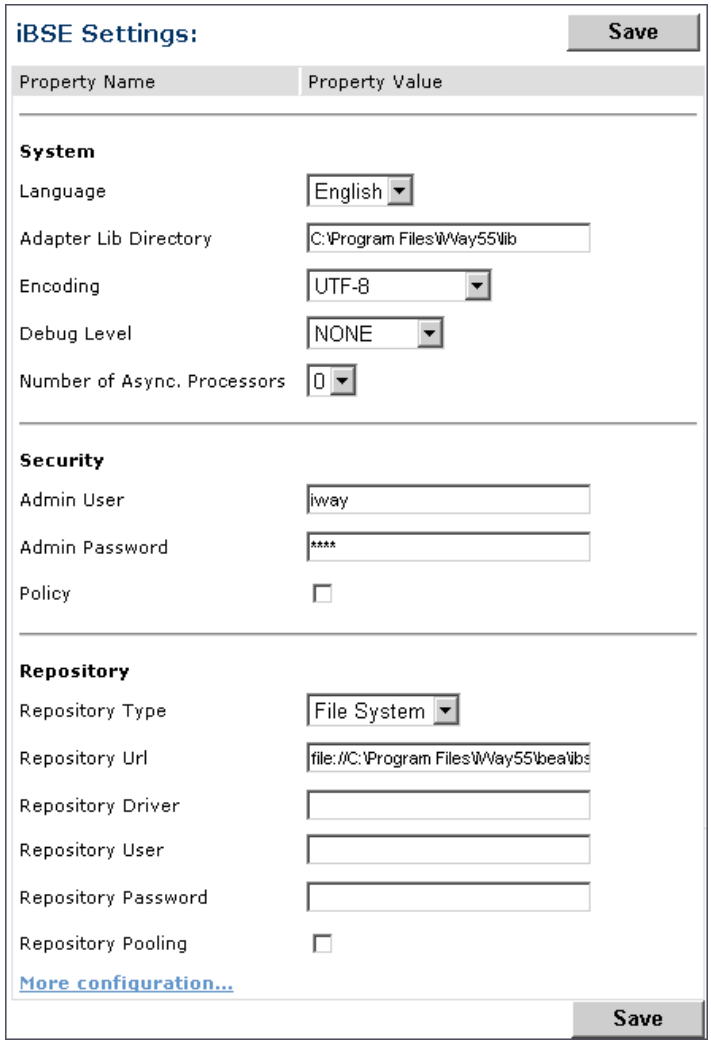
   port

   Is the HTTP port for the application server.

   The iBSE Settings window opens as shown in the following image. It consists of three panes. To configure system settings, the System pane includes 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 adapter 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 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 access more configuration settings.



3. Click *More configuration*.

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

`http://hostname:port/ibse/IBSEStatus`

where:

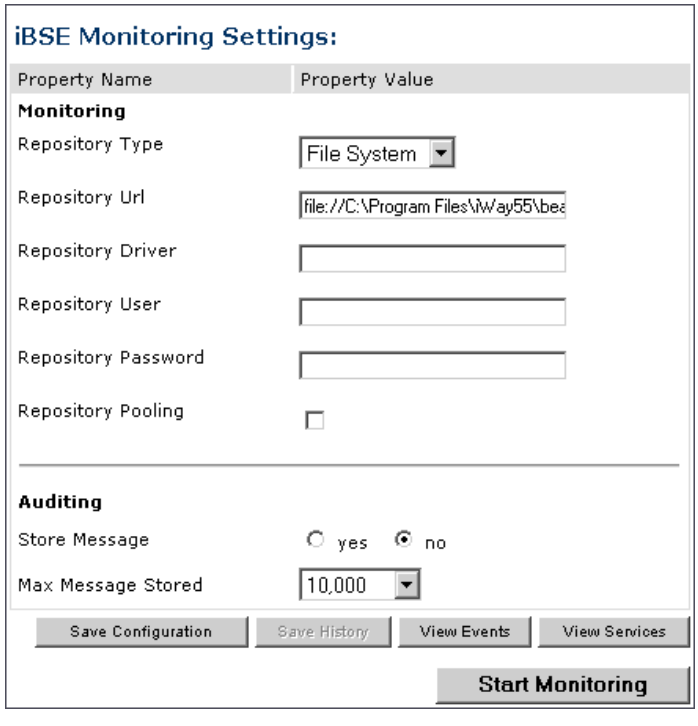
hostname

Is the machine where the application server is running.

port

Is the HTTP port for the application server.

The iBSE Monitoring Settings window which is divided into two panes opens as shown in the following image. At the bottom of the window is a row of command buttons that enable you to save your configuration, view events, or view services. The Save History button is inactive.



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, select the *Repository Pooling* check box.

f. In the Auditing pane, click *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. From the drop-down list, select the maximum number of messages 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 require 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.

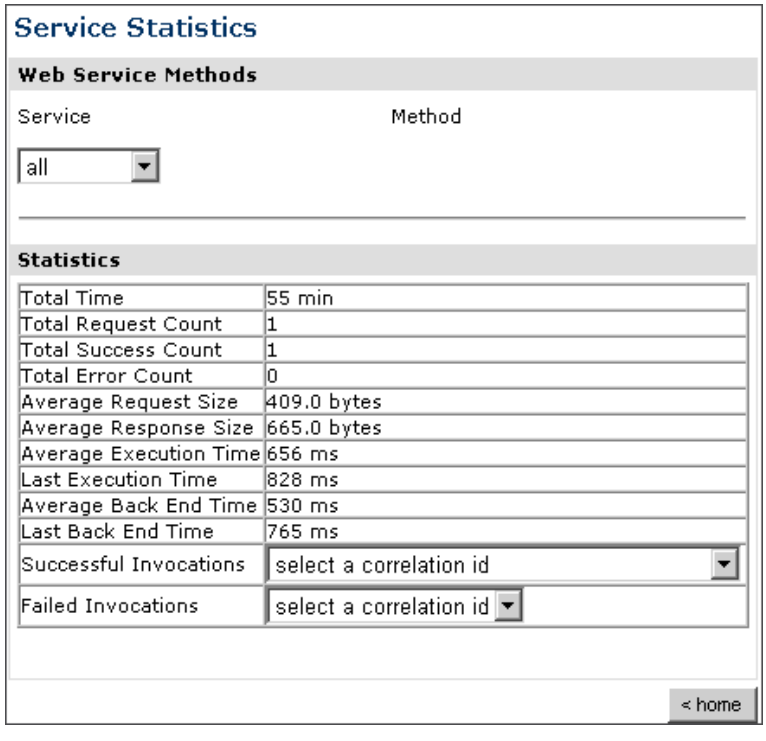
5. To stop monitoring, click *Stop Monitoring*.

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 System Level Summary (Service Statistics) window opens, as shown in the following image. 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.



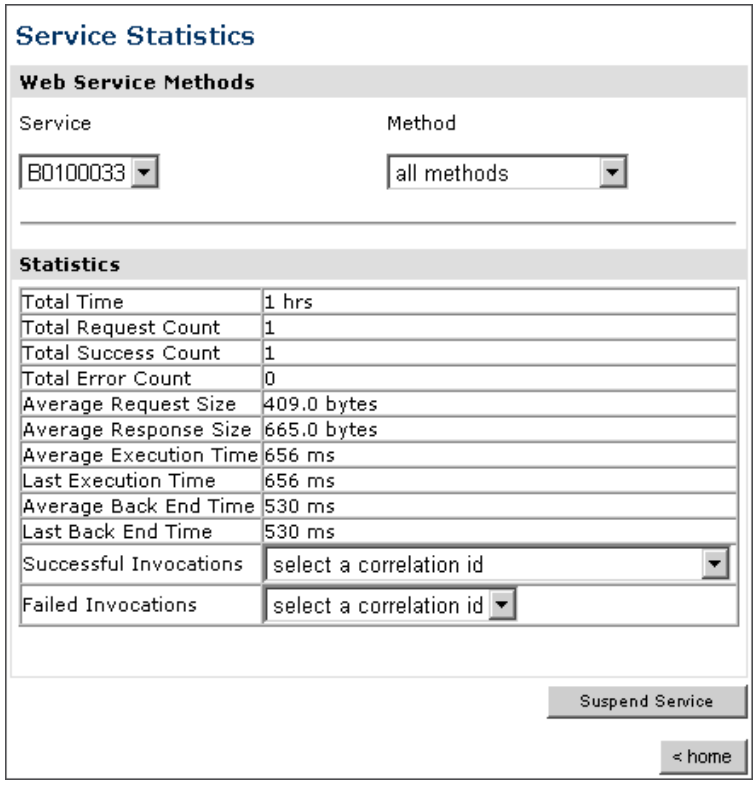
   The system level summary provides services statistics at a system level.

The following table lists and describes each 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 made during the monitoring session.
Total Success Count	Total number of successful service executions.
Total Error Count	Total number of errors encountered.
Average Request Size	Average size of an available service request.
Average Response Size	Average size of an available service response.
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	Successful services arranged by correlation ID. To retrieve more information for a service, select the service from the drop-down list.
Failed Invocations	Failed services arranged by correlation ID. To retrieve more information for a service, select the service from the drop-down list.

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

The System Level Summary (Service Statistics) window opens as shown in the following image. 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.

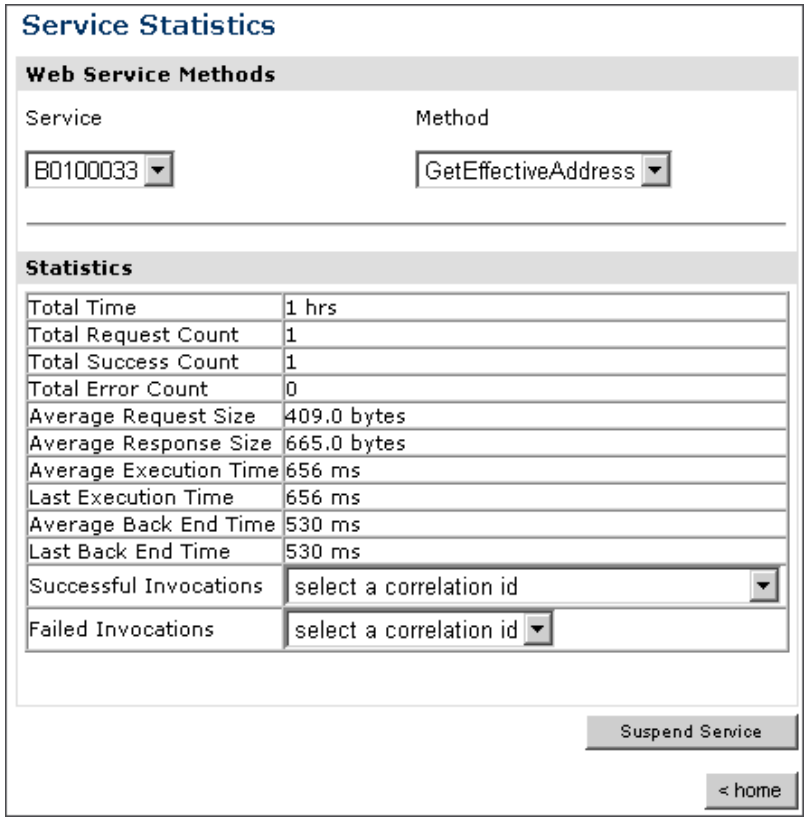


a. To stop a service at any time, click *Suspend Service*.

b. To restart the service, click *Resume Service*.

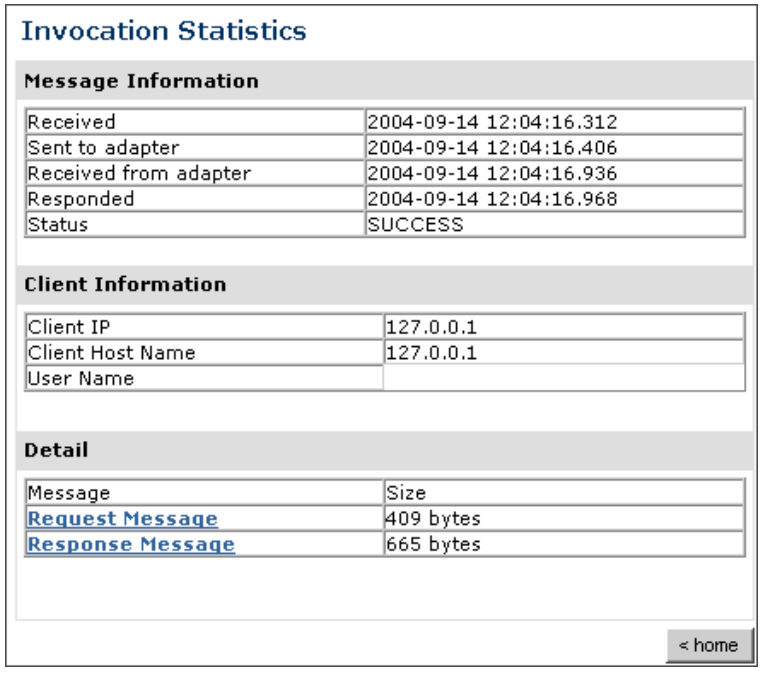
5. Select a method for the service from the Method drop-down list.

The Method Level Summary (Service Statistics) window opens as shown in the following image. 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.



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 Invocation Level Statistics window opens as shown in the following image. 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.



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.

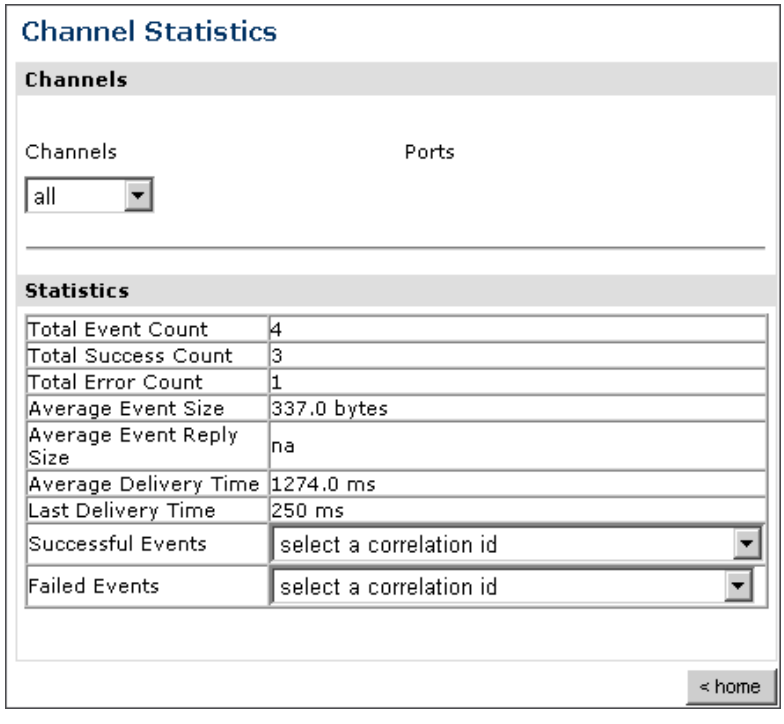
8. To return to the iBSE Monitoring Settings window, click *home*.

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 System Level Summary (Channel Statistics) window opens as shown in the following image. 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.



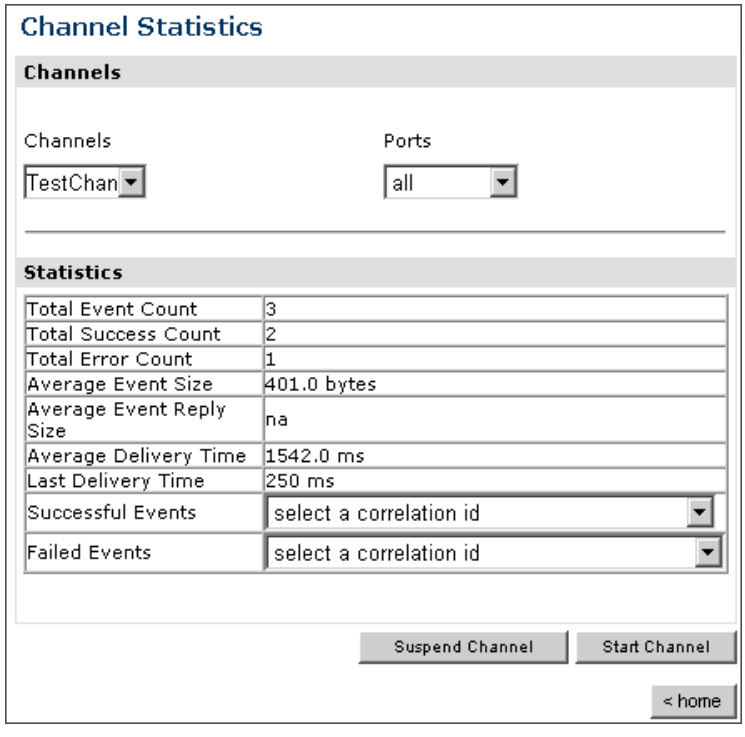
   The system level summary provides event statistics at a system level.

The following table lists and describes each event statistic.

Statistic	**Description**
Total Event Count	Total number of events.
Total Success Count	Total number of successful event executions.
Total Error Count	Total number of errors 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	Successful events arranged by correlation ID. To retrieve more information for an event, select the event from the drop-down list.
Failed Events	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 Channel Level Event Summary (Channel Statistics) window opens as shown in the following image. 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.

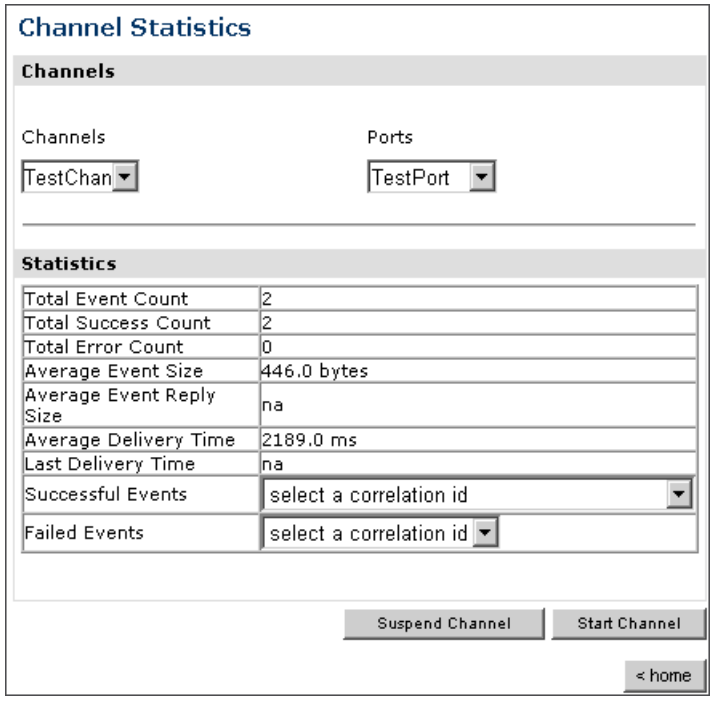


   a. To stop a channel at any time, click *Suspend Channel*.

   b. To start the channel, click *Start Channel*.

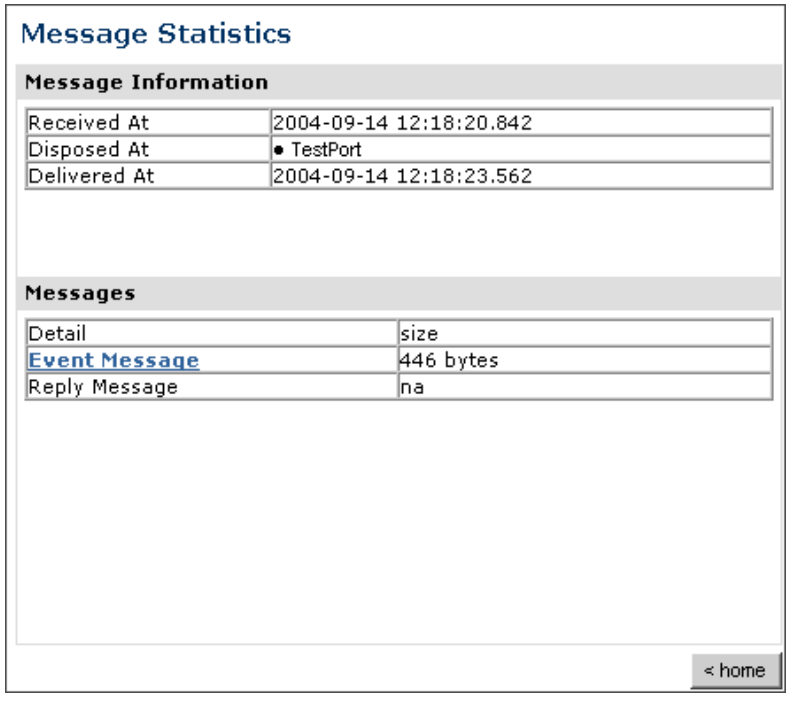
5. From the Ports drop-down list, select a port for the channel.

The Port Level Event Summary (Channel Statistics) window opens as shown in the following image. 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.



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 Event Level Statistics (Message Statistics) window opens as shown in the following image. 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.



   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. The console also displays 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://hostname:port/iwjcaivp`

   where:

   hostname

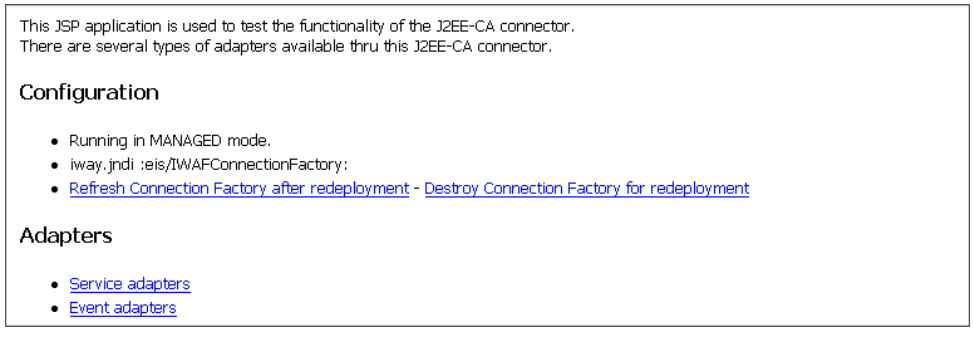
   Is the name of the machine where your application server is running.

   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 pane that opens. The pane 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.

   c. In the JCA Test Tool, click *Refresh Connection Factory after redeployment*.

3. Click *Service adapters*.

4. Select a service adapter to monitor.

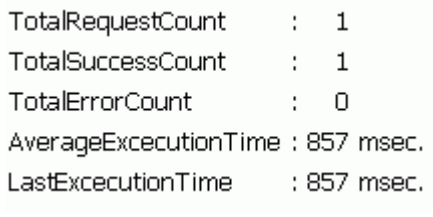
   a. Click the desired target for your service adapter.

   b. In the Request area, enter a user name and password.

   c. In the Input Doc area, enter a request document 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.



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://hostname:port/iwjcaivp`

   where:

   hostname

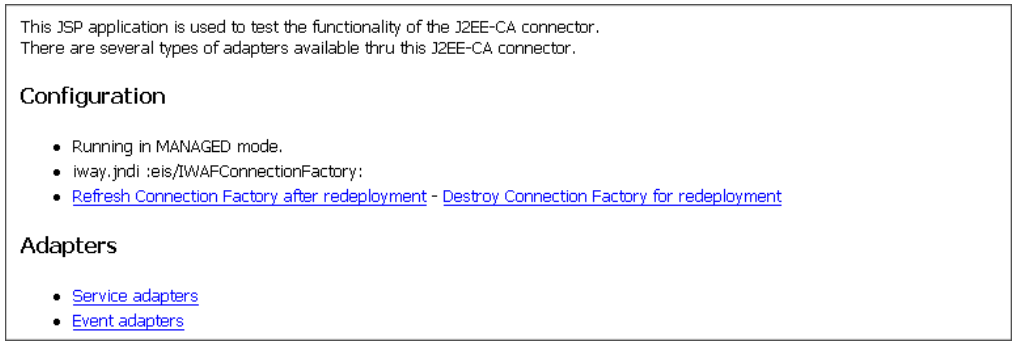
   Is the name of the machine where your application server is running.

   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 JCA Test Tool pane opens as shown in the following image. The pane 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. To monitor the latest event adapter configuration, perform the following steps.

   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.

   c. In the JCA Test Tool, click *Refresh Connection Factory after redeployment*.

3. Click *Event adapters*.

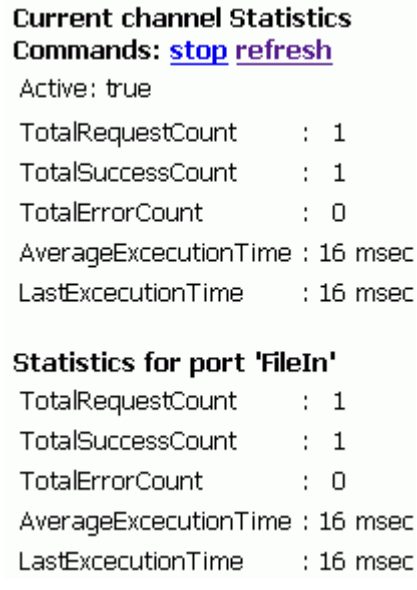
   The Event Adapters pane opens.

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. The upper right of the pane contains options to start or refresh the channel.



Setting Engine Log Levels

The following procedures describe how to set engine log levels. For more information, see the *BEA WebLogic ERP Adapter Installation and Configuration* manual.

Procedure: How to Enable Tracing for Servlet iBSE

To enable tracing for Servlet iBSE:

1. Open the Servlet iBSE configuration window at:

   `http://hostname:port/ibse/IBSEConfig`

   where:

   hostname

   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. Extract the ra.xml file from the iwafjca.rar file.
2. Open the extracted ra.xml file in a text editor.
3. 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>
   ```

   Leave the remainder of the file unchanged.
4. Save the file and exit the editor.
5. Add the file back to the iwafjca.rar file.
6. Redeploy the connector.

A directory in the configuration directory contains the logs. For example, on Windows the default location is the following:

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

The log files are named jca_*.log.

You should also review the logs generated by your application server.

Configuring Connection Pool Sizes

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

Procedure: How to Configure Connection Pool Sizes

To configure connection pool sizes:

1. Extract the ra.xml file from the iwafjca.rar file.
2. Open the extracted ra.xml file in a text editor.
3. 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>
```

4. Save the file and exit the editor.
5. Return the ra.xml file to the iwafjca.rar file.
6. Redeploy the connector.

Migrating Repositories

During design time, a repository is used to store metadata that is created when using Application Explorer to:

- Configure adapter connections.
- Browse EIS objects.
- Configure services.
- Configure listeners to listen for EIS events.

For more information on configuring repositories, see the *BEA WebLogic ERP Adapter Installation and Configuration* manual.

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 can migrate a repository from a development environment to a production environment. The BEA WebLogic Server must be restarted to detect new repository changes.

For more information on migrating repositories, see the *iWay 5.5.011 for BEA WebLogic 9.1 Migration Guide*.

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

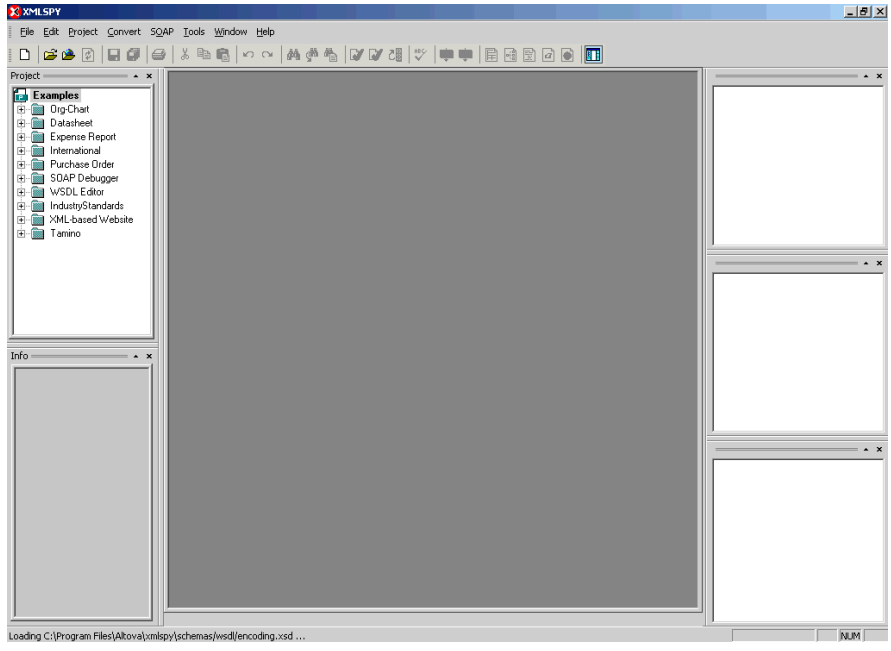
To export a target:

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.

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

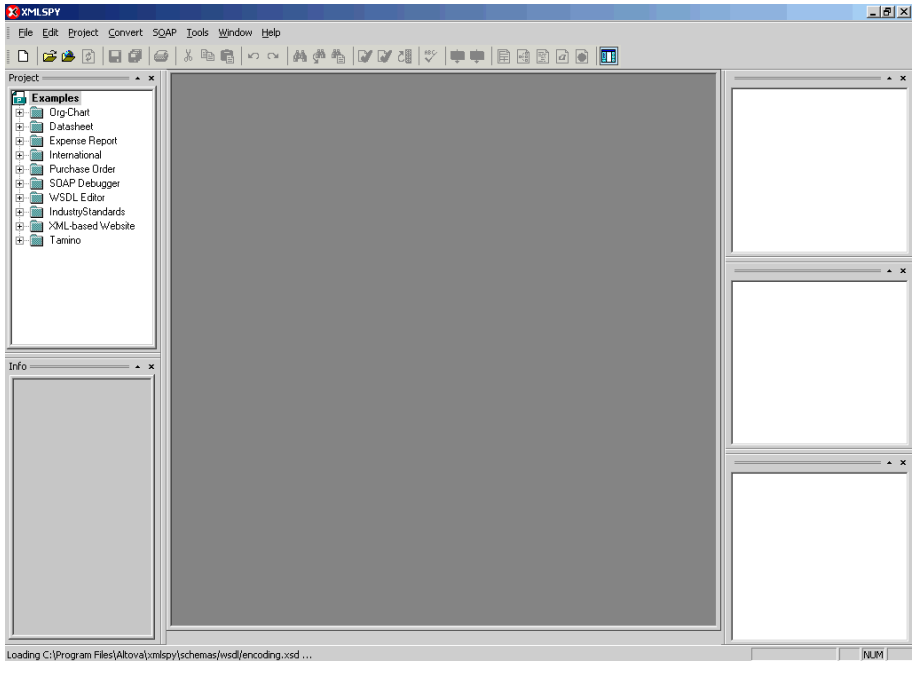
To import a target:

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 tool bar.
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>R0lGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</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

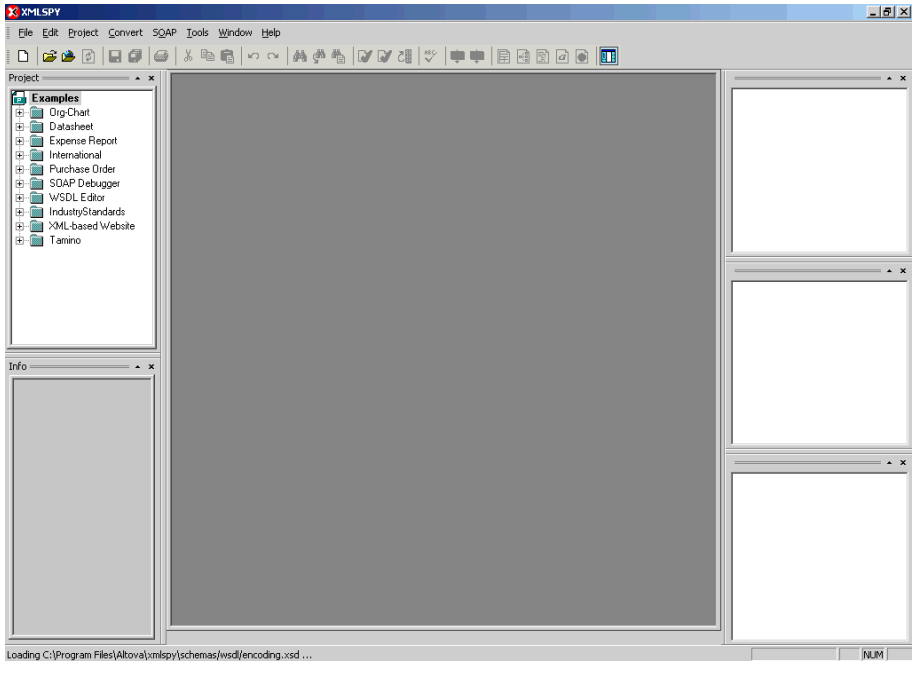
To retrieve Web service method connection information:

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 tool bar.

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: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

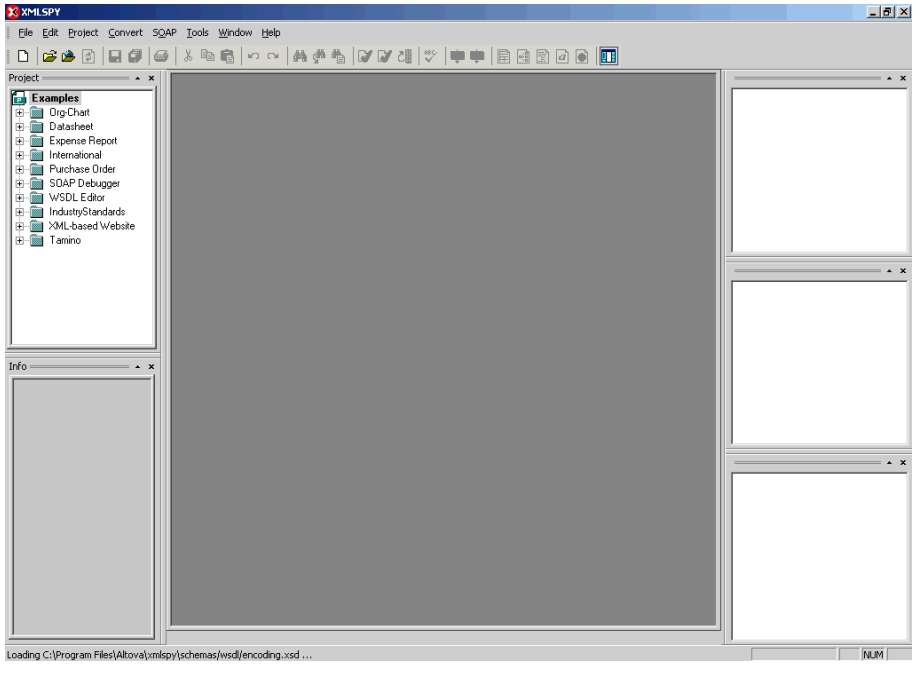
To update Web service method connection information:

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 tool bar.
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: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

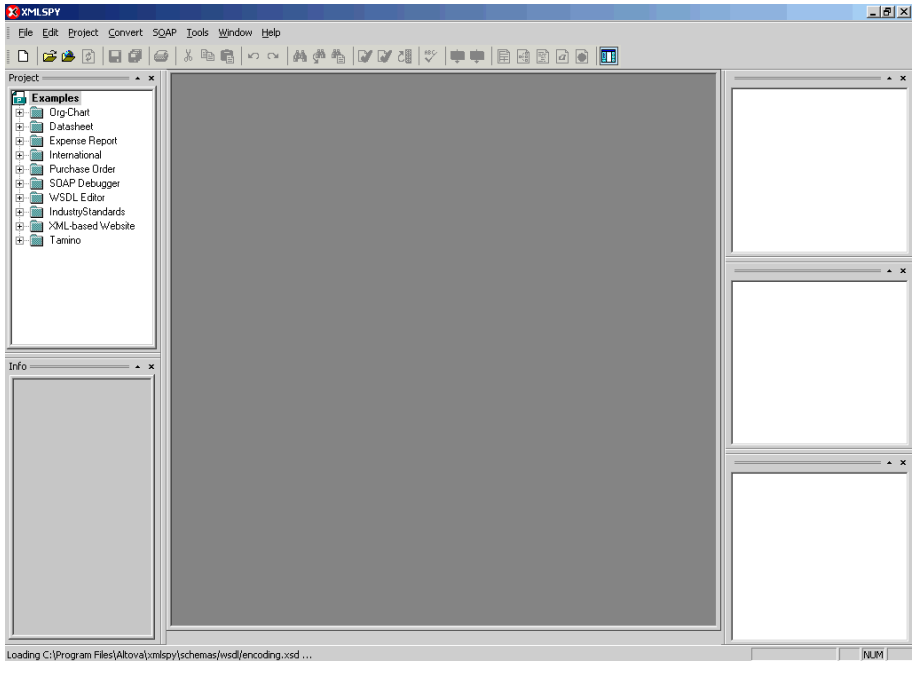
To start a channel programmatically:

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.

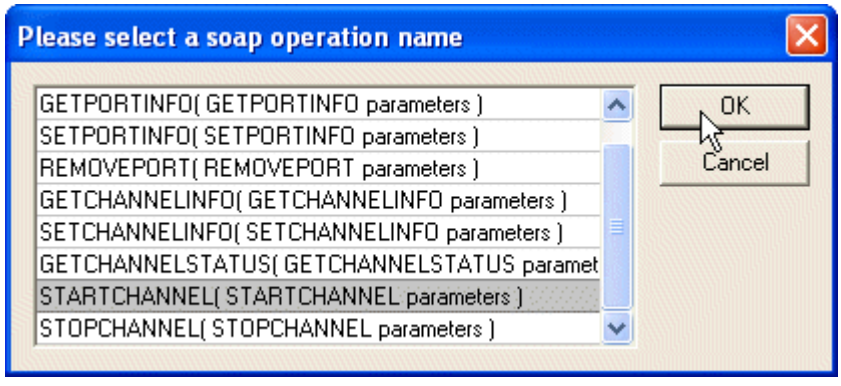


3. From the SOAP menu, select *Create new SOAP request*.

   The WSDL file location dialog box opens.

4. In the Choose a file field, paste the iBSE control event URL, and click *OK*.

   The following image shows the soap operation name dialog box that opens with a list of available control methods.



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 tool bar.

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

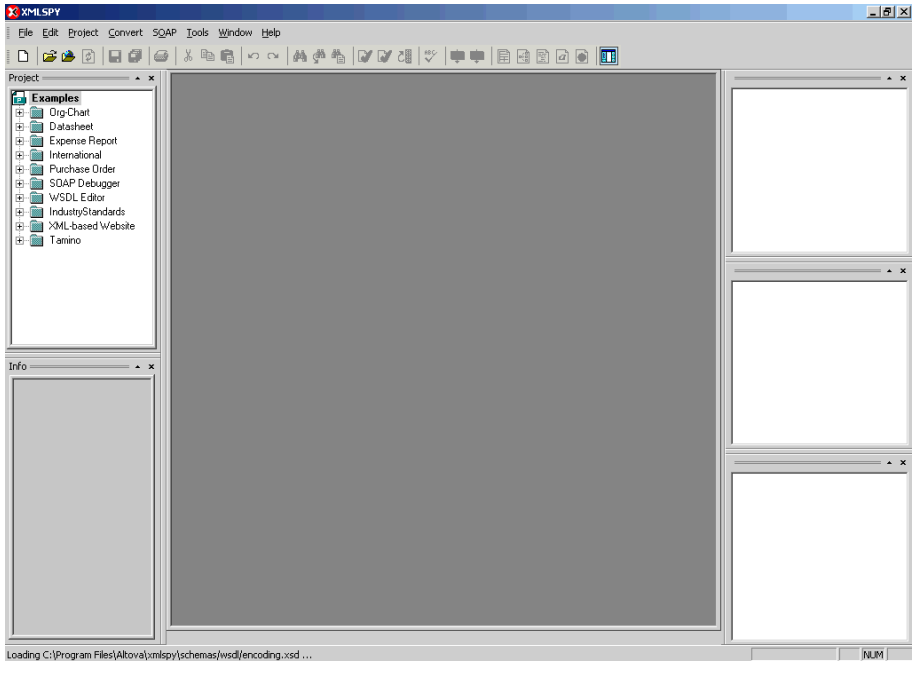
To stop a channel programmatically:

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.

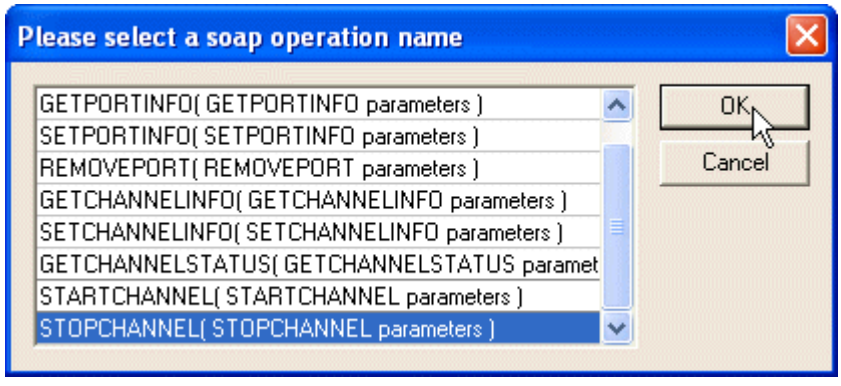


3. From the SOAP menu, select *Create new SOAP request*.

   The WSDL file location dialog box opens.

4. In the Choose a file field, paste the iBSE control event URL, and click *OK*.

   The following image shows the soap operation name dialog box that opens with a list of available control methods.



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 tool bar.

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 7

Understanding SAP Events

Topics:

- Overview
- Related Concepts and Terminology
- Registering Your Program ID in SAPGUI
- Testing the SAP Event Adapter
- Application Link Enabling Configuration for the Event Adapter
- Testing the SAP ALE Configuration

The following topics provide an overview of event functionality in SAP and describe how to configure and test your SAP system for event processing.

Overview

An event in SAP is defined as an occurrence of a status change in an object. The event is created when the relevant status change occurs. You or SAP must implement event creation.

An event is created from a specific application program (the event creator) and then "published" system-wide. An unlimited number of receivers can respond to the event with their own "response mechanisms". An event is usually defined as a component of an object type.

SAP "pseudo events" are events that are not processed by the SAP Event manager, but are called from an ABAP program or Remote Function call (using the Destination parameter).

Related Concepts and Terminology

The following topics list and define specific terminology related to SAP and SAP event handling.

Client and Server Programs

RFC (Remote Function Call) programs for non-SAP systems can function as either the caller or the called program in an RFC communication. The two types of RFC programs are:

- RFC Client
- RFC Server

The RFC client is the instance that calls the RFC to execute the function that is provided by an RFC server. The functions that can be executed remotely are called RFC functions, and the functions provided by the RFC API are called RFC calls.

SAP Gateway

The SAP Gateway is a secure BEA WebLogic Server. No connections are accepted unless they were pre-registered previously from the SAP presentation Client. A server connection presents itself to the Gateway and exposes a Program Identifier. If the Program Identifier is found in the list of registered Program IDs, the Gateway server then offers a connection to the server, which accepts a connection.

The Program ID then is linked with an RFC Destination within SAP, which enables SAP Function Modules and ALE documents (IDocs or BAPI IDocs) to be routed to the destination. The RFC Destination functions as a tag to mask the Program ID to SAP users.

An RFC server program can be registered with the SAP Gateway and wait for incoming RFC call requests. An RFC server program registers itself under a Program ID at an SAP Gateway and not for a specific SAP system.

In SAPGUI, the destination must be defined with transaction SM59, using connection type T and Register Mode. Moreover, this entry must contain information on the SAP Gateway where the RFC server program is registered.

Program IDs and Load Balancing

If the Gateway Server has a connection to a particular server instance and another server instance presents itself to the Gateway, the Gateway offers the connection and then begins functioning in Load Balancing mode. Using a proprietary algorithm, the Gateway sends different messages to each server depending on demand and total processing time. This could cause unpredictable results in the BEA WebLogic scenario where messages are validated by schema and application.

When configuring multiple events in BEA WebLogic using a single SAP program ID, SAP load balances the event data. For example, if multiple remote function calls or BAPIs use the same program ID (for example, IWAYID) and multiple SAP listeners are configured with this program ID, then SAP sends one request to one listener and the next to another listener, and so on.

The SAP Gateway Server includes a load balancing algorithm. This mechanism is proprietary to SAP application development and may work by comparing total throughput of the connection, the number of times in wait state, and so on. This means connection 1 may receive nine messages and connection 2 may receive one message. If five of nine messages are rejected for schema validation and the message on the other ID is rejected for schema validation, the customer can very easily make a case of missing messages.

Registering Your Program ID in SAPGUI

To enable your SAP system to issue the following calls or interfaces to the SAP event adapter, you must register your program ID under an RFC destination.

- Remote Function Calls (RFC)
- Business Application Programming Interfaces (BAPI)
- Intermediate Documents (IDoc)

The RFC destination is a symbolic name (for example, IWAYDEST) that is used to direct events to a target system, masking the program ID. The Program ID is configured in both SAPGUI and the event adapter.

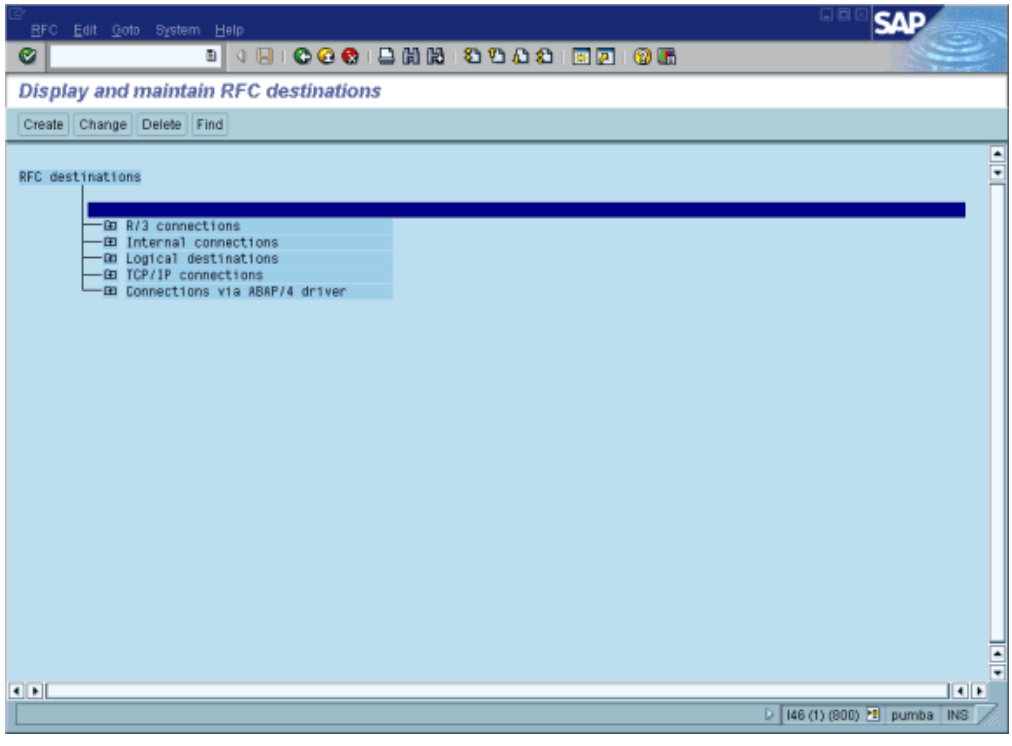
Procedure: How to Register Your Program ID

To register your program ID:

1. Launch the SAP Workbench and logon to the SAP system.
2. Select *Tools*, *Administration*, *Network*, and then *RFC destination*.

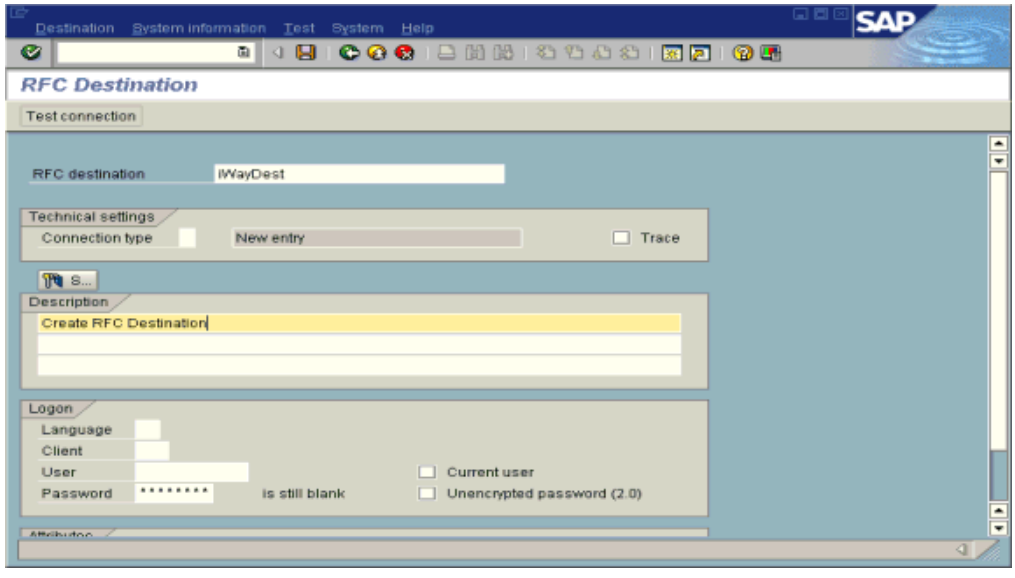
3. Execute the *SM59* transaction.

   The Display and maintain RFC destinations window opens and displays a list of connections and drivers you can manage as shown in the following image.



4. Select *TCP/IP connections*.
5. Click *Create*.

The RFC Destination window opens and displays fields where you provide information about the RFC destination as shown in the following image.



a. In the RFC destination field, type a name, for example, IWAYDEST.

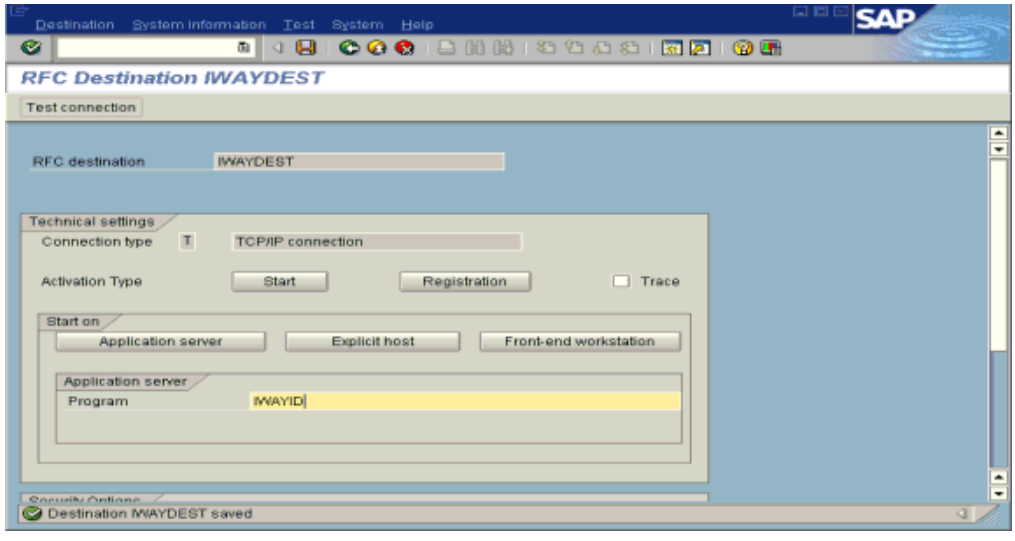
The value you type in this field is case-sensitive.

b. In the Connection type field, type *T* (for destination type, TCP/IP).

c. In the Description field, type a brief description.

6. Click *Save* from the tool bar or select *Save* from the Destination menu.

The RFC Destination IWAYDEST window opens as shown in the following image.



a. For the Activation Type, click the *Registration* button.

b. In the Program field, type *IWAYID*.

7. Click *Save* from the tool bar or select *Save* from the Destination menu.

8. Ensure your event adapter is running.

9. To verify that the SAP system and the BEA WebLogic Adapter for SAP are communicating, click *Test connection*.

Testing the SAP Event Adapter

In the SAP Server, the SE37 transaction enables you to send RFCs (Remote Function Calls) or BAPIs (Business Application Programming Interfaces) to any RFC destination. For more information on RFC destinations, see *Registering Your Program ID in SAPGUI* on page 7-3.

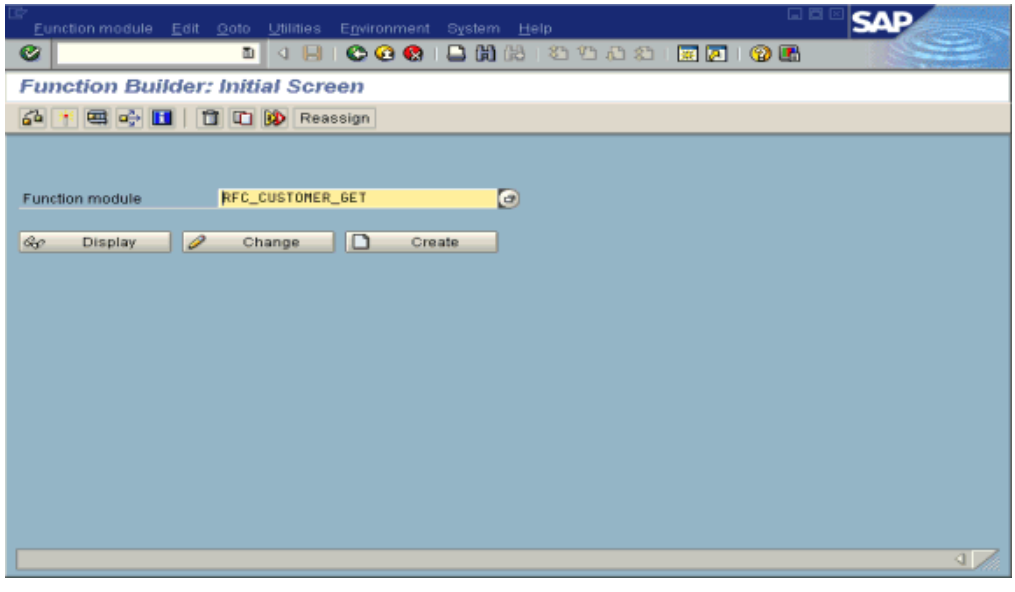
Note: Depending on the release or service pack installed, certain RFCs may not exist in your particular SAP system. Therefore, the examples included in this documentation may not be relevant to your system. If this is the case, you should use the examples as a general reference for adapter functionality and choose an RFC that exists within your SAP application environment.

Procedure: How to Test the SAP Event Adapter by Sending RFCs or BAPIs Manually

To test the SAP event adapter:

1. In the Function Builder: Initial Screen, select a function module, for example, RFC_CUSTOMER_GET.

   The following image shows the Function Builder: Initial Screen where you can select to display, change, or create a function module. RFC_CUSTOMER_GET is selected.



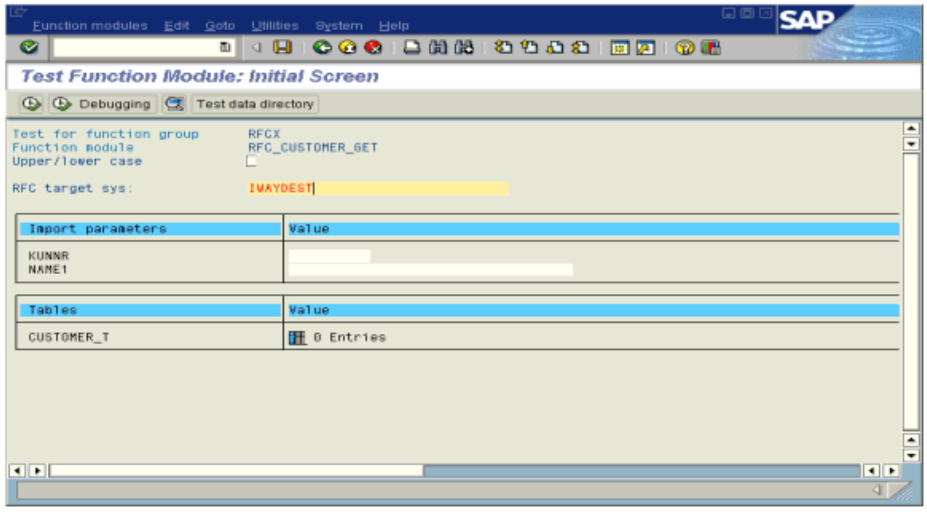
   a. To choose single test, press *F8* and click the *Single Test* icon or select *Function module*, *Test,* and then *Single Test*.

   b. Enter an RFC target system, for example, IWAYDEST.

   c. Enter input data for the particular RFC module, for example, AB*.

2. To execute, press *F8*.

The Test Function Module: Initial Screen opens as shown in the following image. It includes information about the test, the function module, and the target system. You can select the check box for Upper/lower case. The upper left pane lists the import parameters, and the upper right pane contains fields for the values. The lower left pane lists tables, and the lower right pane lists the number of entries.



3. Enter data into the SAP GUI and click the *Execute* button.

The function name and input data are transferred by RFC to create an XML document on theBEA WebLogic Server with the parameters input in SAPGUI.

Application Link Enabling Configuration for the Event Adapter

The SAP event adapter receives IDocs (Intermediate Documents) from SAP.

Document Processing

The BEA WebLogic Adapter for SAP accepts IDocs in either SAP standard format or XML format. If you are using flat IDocs, no encoding is required and the EDI_DC40 control header must be the first entry in the IDoc.

Collected Inbound Configuration

When sending collected IDocs to SAP, if the DOCNUM field does not have a unique document number for each IDoc, the system creates an IDoc for each header record in the collected IDoc file. The system also duplicates the data for each IDoc.

You must ensure that the DOCNUM field is included in the EDI_DC40 structure and that each IDoc has a unique sequence number within the collected IDoc file.

To configure an SAP system to send IDocs to the SAP event adapter, you use the ALE (Application Link Enabling) configuration to:

1. Register your program ID in SAPGUI. For more information, see *Registering Your Program ID in SAPGUI* on page 7-3.
2. Define a port.

   A port identifies where to send messages. The port can be used only if an RFC destination was previously created.

   For more information on creating an RFC destination, see *Overview* on page 7-2. For more information on defining a port, see *How to Define a Port* on page 7-9.
3. Create a logical system.

   One type of partner is a logical system. A logical system manages one or more RFC destinations. For more information, see *How to Create a Logical System* on page 7-10.
4. Create a partner profile.

   A partner profile is a definition of parameters for the electronic interchange of data with a trading partner using the IDoc interface. To communicate with a partner using the IDoc interface, you must create a partner profile. For more information, see *How to Create a Partner Profile* on page 7-12.
5. Create a distribution model for the partner and message type.

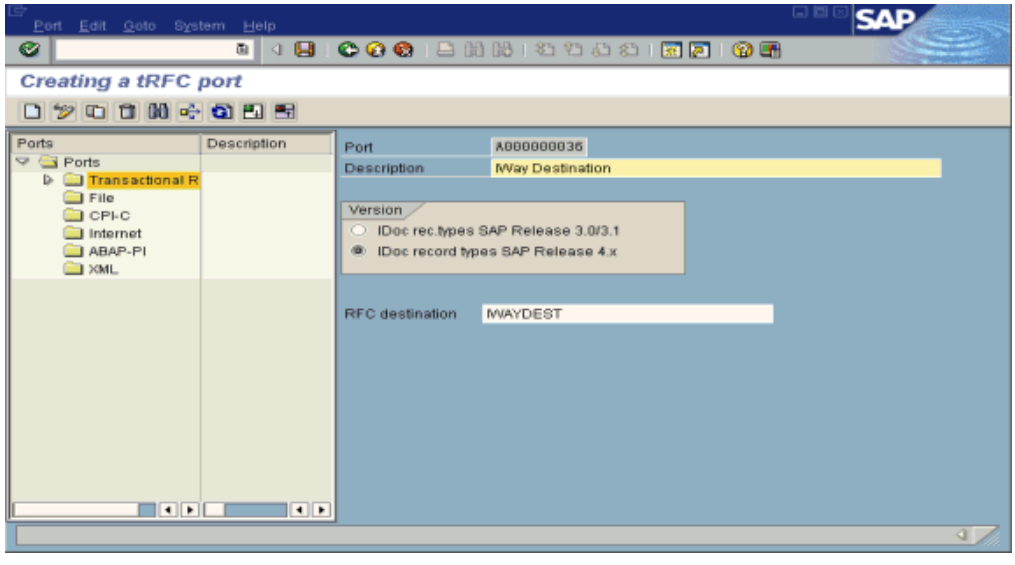
   You create a distribution model for the partner and message type you designated. For more information, see *How to Create a Distribution Model for the Partner and Message Type* on page 7-14.
6. Test the SAP event adapter. For more information, see *Testing the SAP ALE Configuration* on page 7-17.

Procedure: How to Define a Port

To define a port:

1. In the ALE configuration, choose *Tools, Business Communications, IDocs Basis, IDoc,* and then *Port Definition* or execute the *WE21* transaction.

The Creating a tRFC port window opens as shown in the following image. On the left, the window is divided into a Ports pane and a Description pane. A pane for displaying information about the port is on the right.



a. In the left pane under Ports, select *Transactional RFC* and click *Create*.

b. Select *Generate port name*.

The system generates the port name.

c. In the right pane, select the IDoc version you want to send through this port.

d. Click the destination you created, for example, IWAYDEST.

2. Save the session, making note of the system-generated RFC port.

Procedure: How to Create a Logical System

To create a logical system called IWAYLOG:

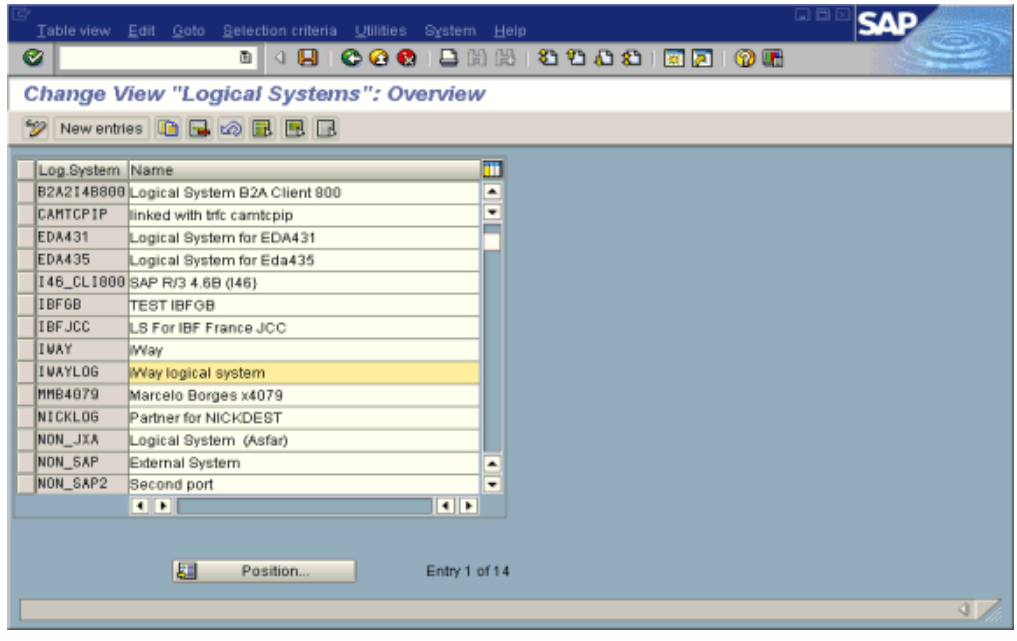
1. In the ALE Configuration, enter the area menu selection *SALE* transaction.

2. Select *SAP Reference IMG*.

3. Expand the following nodes: *Basis Components, Application Link Enabling (ALE), Sending and Receiving Systems, Logical Systems*, and *Define Logical System*.

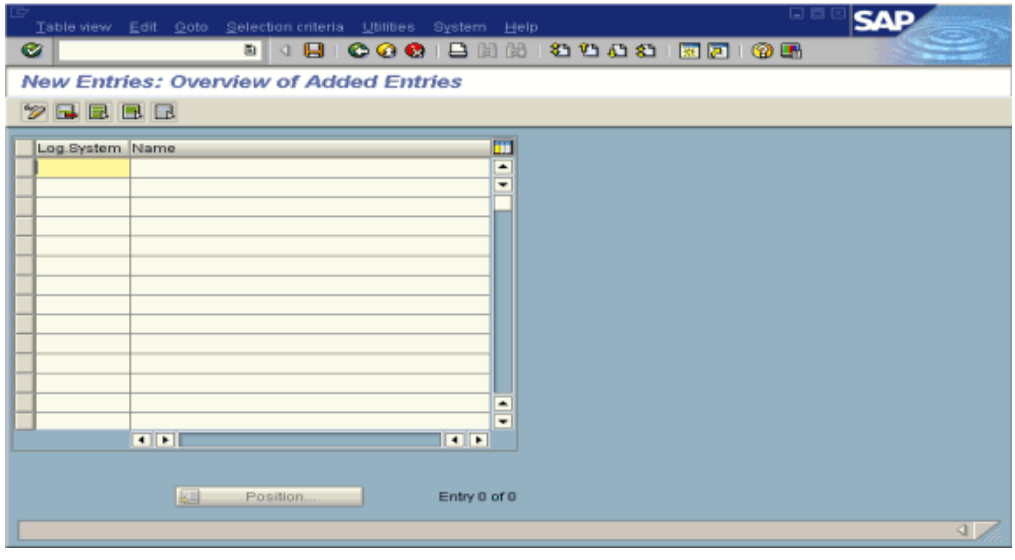
4. Click the green check mark beside *Define Logical System*.

The Change View "Logical Systems": Overview window opens and displays a list of logical systems and their names as shown in the following image.



5. Click *New entries*.

The New Entries: Overview of Added Entries window opens, as shown in the following image, with columns labelled Log.System and Name for adding new log systems.



a. Type an entry for Log System, for example, IWAYLOG.

b. In the Name column, type a name (description) for the partner profile.

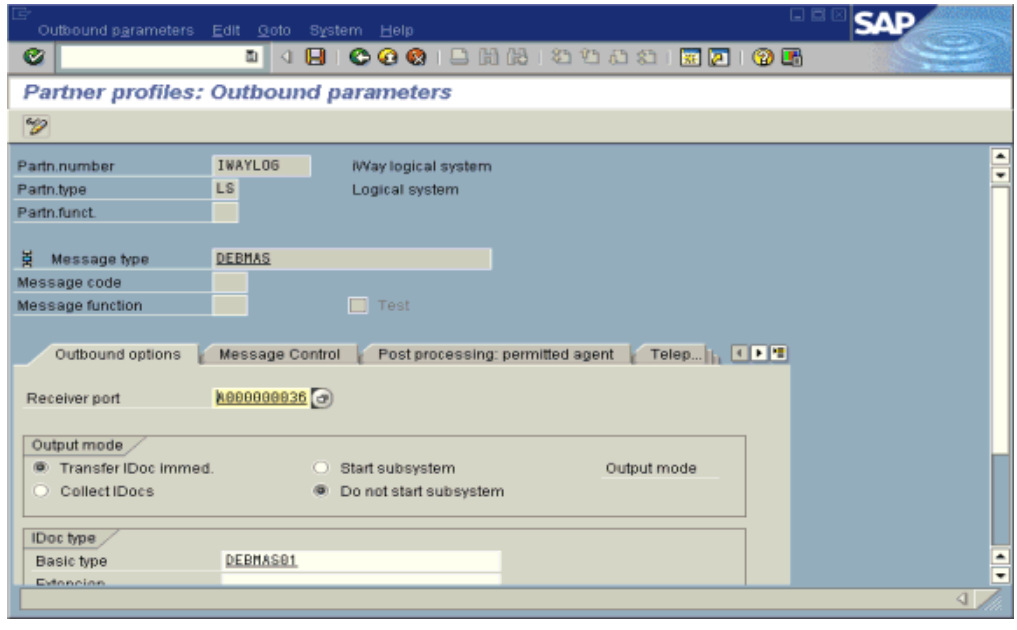
6. Save the session.

Procedure: How to Create a Partner Profile

To create a partner profile:

1. In the SAP Workbench, choose *Tools, Business Communication, IDoc Basis, IDoc,* and then *Partner profile* or execute the *WE20* transaction.

The Partner profiles: Outbound parameters window opens and displays fields for specifying details for the partner profile as shown in the following image.



a. Select Partner type *LS* (Logical system).

b. Press *F5* (Create).

2. For Type, enter *USER*.

3. For Agent, enter the current user ID, or you may select another agent type.

4. Under the outbound parameter table control, select *Create outbound parameter*.

Partn.type is LS.

Message type is DEBMAS (the IDoc document type).

5. Leave *Partn.funct* blank.

6. Click the *Outbound options* tab.

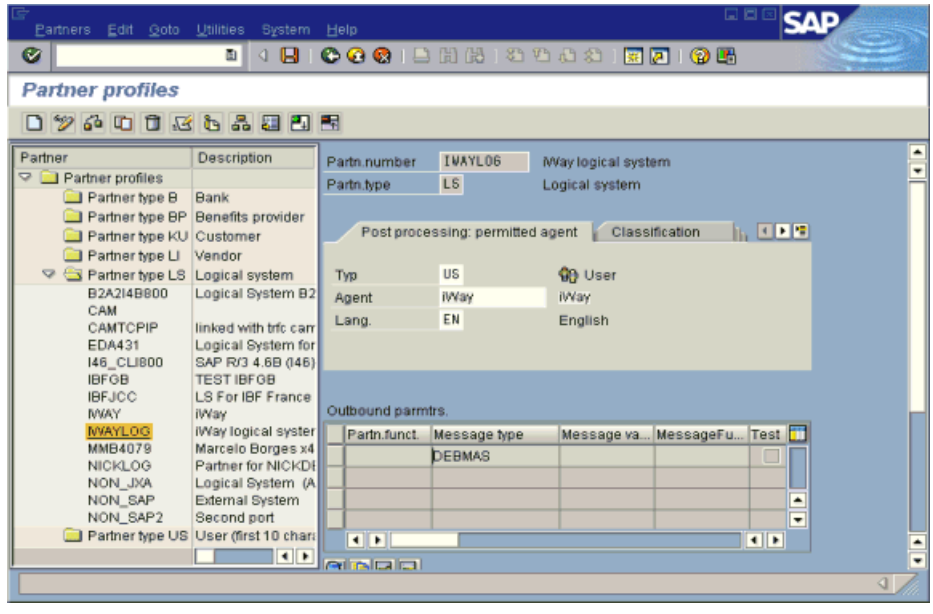
a. Depending on your performance requirements, click *Transfer IDoc Immed* or *Collect IDocs*.

b. For the IDoc, type a message type, for example, DEBMAS.

c. Type a receiver port, for example, A000000036.

7. Save the session and exit.

The Partner profiles summary window opens and displays information for the logical system that you created as shown in the following image. In the left pane are partners and descriptions. The right pane displays information depending on which tab is active.



Procedure: How to Create a Distribution Model for the Partner and Message Type

To create a distribution model called IWAYMOD:

1. In the SAP Workbench, choose *Tools, AcceleratedSAP, Customizing,* and then *Project Management* or execute the *BD64* transaction.

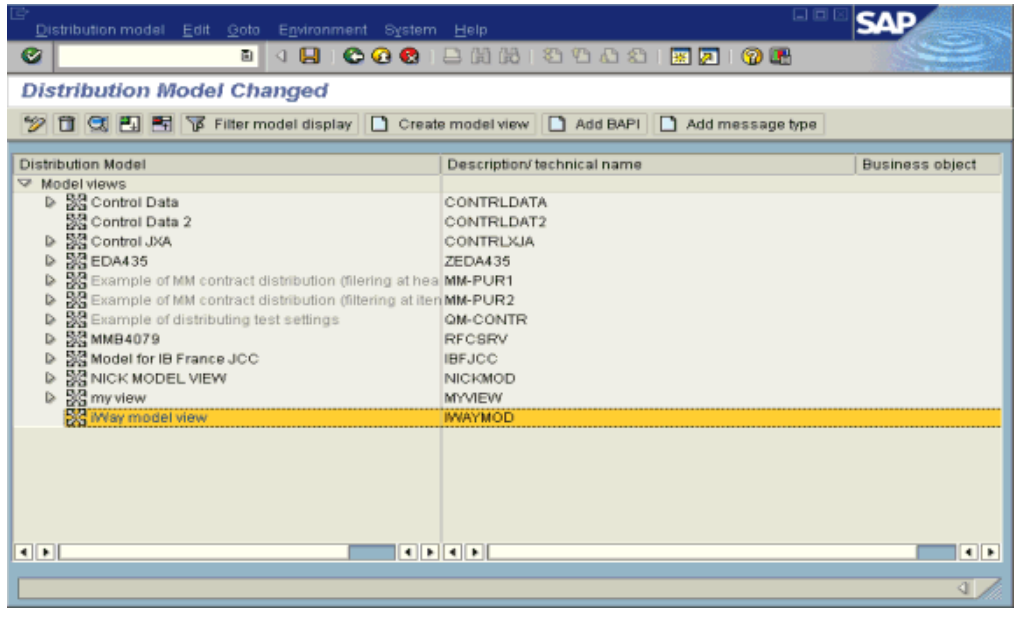
   The Display Distribution Model window opens.

2. Select *Create model view.* (If required, switch processing mode to edit within Distribution Model/Switch Processing Mode.)

3. Type a short text string and a technical name for your new model view.

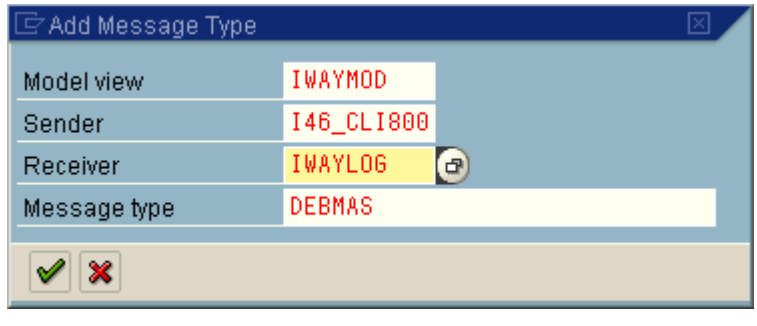
4. Click the *Save* button.

The Distribution Model Changed window opens with a tree structure of the distribution model in the left pane and the descriptions or technical names in the right pane, as shown in the following image.



5. In the Distribution Model tree, select a new model view.
6. At the right, in the button bar, select *Add message type*.

   The Add Message Type pane opens and displays the name of the model view. It includes fields for specifying the sender and receiver of the message, as well as the message type, as shown in the following image.



   a. In the Sender field, provide the sender that points to the SAP system that sends the IDoc, for example, I46_CLI800.

      In this case, the sender is an SAP 4.6B system.

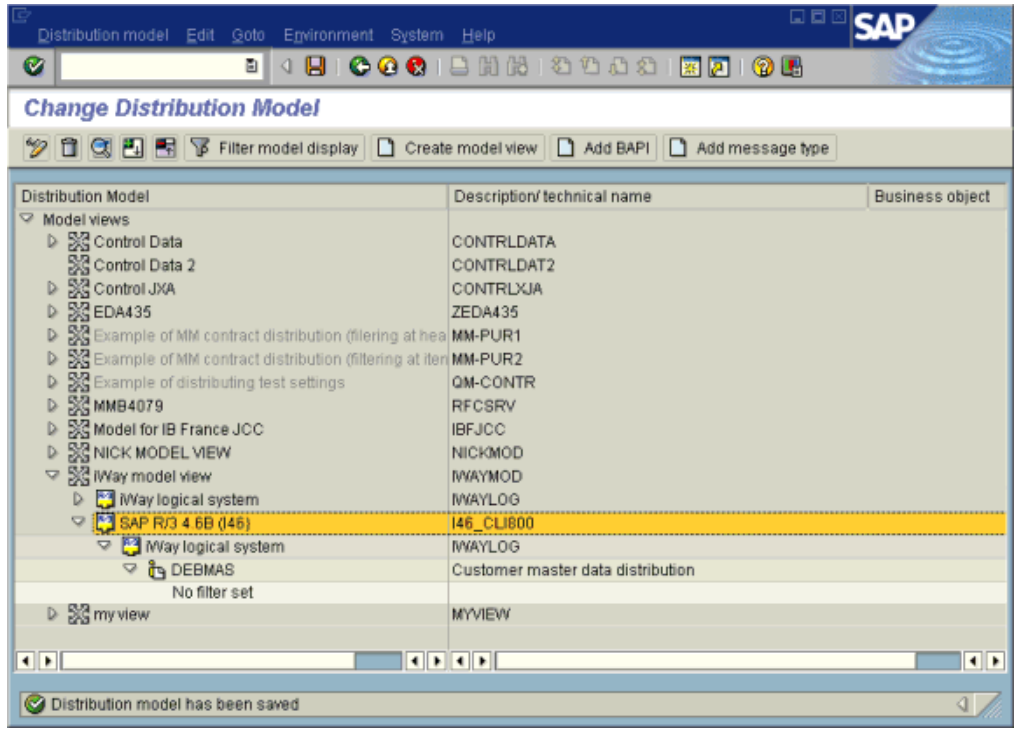
b. In the Receiver field, provide the logical system, for example, IWAYLOG.

c. In the Message type field, provide the type of IDoc, for example, DEBMAS.

7. Click the check mark icon.

8. Click the *Save* button.

The Change Distribution Model window opens and displays the new model view to use to send message type, DEBMAS, from the I46_CLI800 SAP system to the IWAYLOG logical system, as shown in the following image.



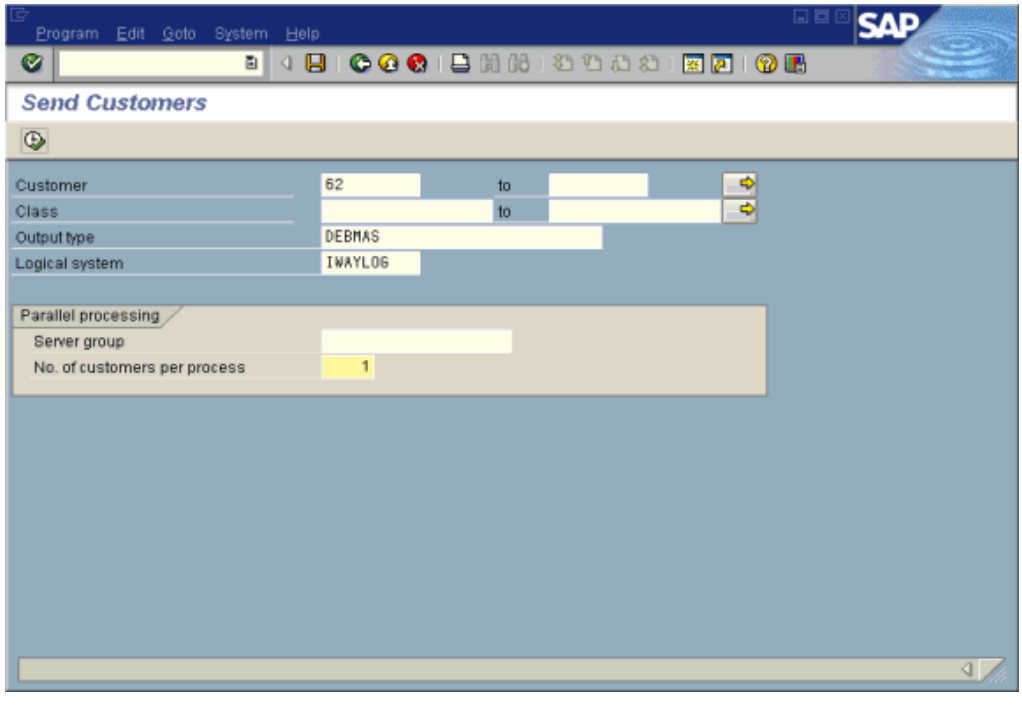
You are now ready to test the connection to the logical system.

Testing the SAP ALE Configuration

In the SAP Server, the BD12 transaction enables you to send IDocs to any logical system, for example, to an event adapter.

Procedure: How to Test the SAP ALE Configuration

The following image shows the Send Customers window where you test the message type. It includes fields for Customer, Class, Output type, and Logical system. The Parallel processing pane includes a field for Server group and a field for the number of customers per process.

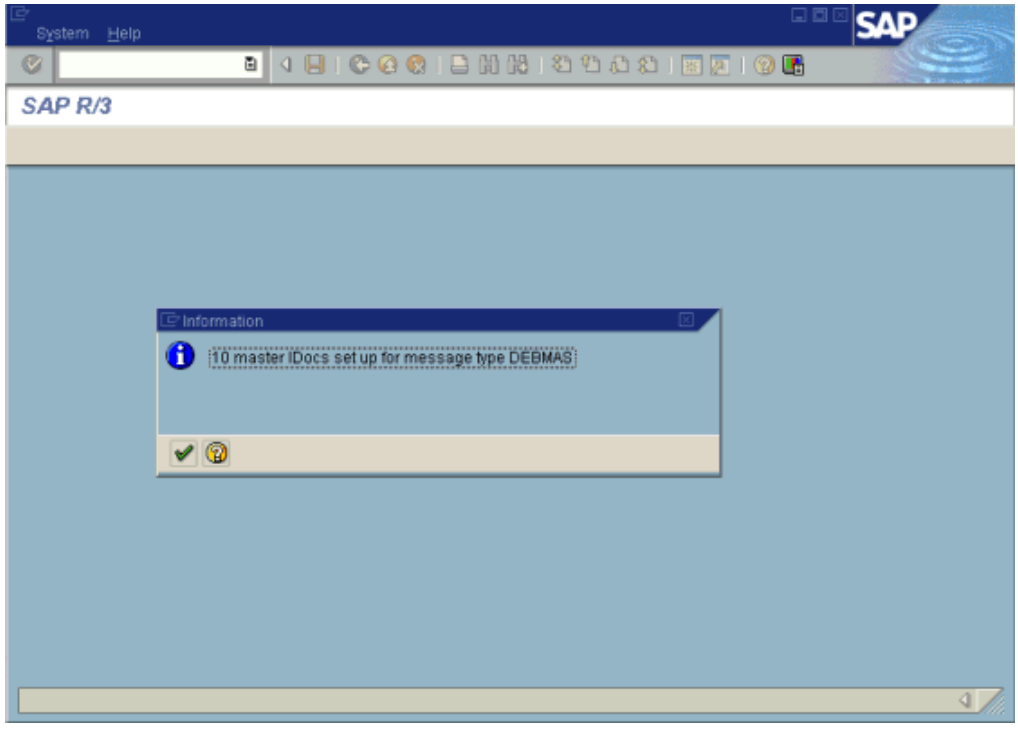


To test the SAP Application Link Enabling (ALE) configuration:

1. In the Send Customers window, type the IDoc message type *DEBMAS* in the Output type field.
2. In the Logical system field, type the logical system, for example, IWAYLOG.
3. To transfer data, click the *Run* button.

   The SAP event adapter receives the IDoc in XML format. No response is expected from the event adapter.

A window opens and confirms the message entered in previous screens, as shown in the following image.



Usage Considerations

The following topic provides insight for the use of SAP event handling in BEA WebLogic.

Multiple Events Using Identical Program IDs

Configuring multiple events in BEA WebLogic that use the same SAP Program ID enables you to load balance SAP event data over multiple adapter event consumers. However, when configuring each event, all the event parameters for each event must match precisely. If one of the event configuration parameters (for example, log level) is different for one of the events, BEA WebLogic creates a different schema for that event, and event data can be lost.

In this scenario, because you are using the same SAP program ID, SAP sends event data to all events configured with that program ID. BEA WebLogic creates separate schema for each event (because configuration parameter(s) differ) and then binds a specific event channel to that event schema. The result is that only events that match that schema are sent over a specific channel. While SAP is sending event data to every event configured with the same program ID, BEA WebLogic, as it validates schemas, rejects any event data that does not match the schema for that channel.

For example, only Doc. A events that appear on the first channel are received, and Doc. B events that appear on the second channel are received. Doc. B events that appear on the first channel are rejected, because they do not pass schema type validation. The same result occurs for Doc. A events that appear on the second channel.

This usage consideration applies to all supported platforms.

Workaround

If you wish to load balance SAP event data over multiple events in BEA WebLogic, ensure you configure each event in precisely the same way.

If you do not wish to use load balancing, configure a separate event for each SAP program ID. Also, ensure that no other event configured in an Application View or in another BEA application uses the same program ID.

CHAPTER 8

Troubleshooting and Error Messages

Topics:

- Troubleshooting Overview
- Error Messages in Application Explorer
- Error Messages in SAP
- Error Messages in JCA
- Error Messages in iBSE

The following topics explain limitations and workarounds when connecting to SAP. The adapter-specific errors described in this section can arise whether you are using the adapter with a JCA or with an iBSE configuration.

This section provides troubleshooting information for SAP for the following four categories:

- Application Explorer
- SAP
- JCA
- iBSE

Troubleshooting Overview

An event in SAP is defined as an occurrence of a status change in an object. Events are created when the relevant status change occurs. Event creation must be implemented by SAP or yourself. An event is created from specific application programs (the event creator) and then "published" system-wide. Any number of receivers can respond to the event with their own response mechanisms. An event is usually defined as a component of an object type.

Note: The following locations include log file information that is relevant for troubleshooting.

- JCA trace information can be found under the following directory:

  ```
  C:\Program Files\iWay55\config\base\log
  ```

- iBSE trace information can be found under the following directory:

  ```
  C:\Program Files\iWay55\bea\ibse\ibselogs
  ```

- The log file for Application Explorer can be found under the following directory:

  ```
  C:\Program File\iWay55\tools\iwae\bin
  ```

Error Messages in Application Explorer

The following table lists errors and solutions when using Application Explorer with the adapter.

Error	Solution
Cannot connect to the adapter from Application Explorer.	Ensure that: SAP is running. The Server name, System Number, and Client Number are correct. The SAP user ID and password are correct.
Cannot connect to the SAP target through Application Explorer. The following error message appears: `Error getting target [SAP] - java.lang.Exception: Error Logon to SAP System`	Ensure that you enter the correct connection parameters when connecting to the SAP target.

Error	**Solution**
SAP does not appear in the Application Explorer adapter node list.	Ensure that you added the sapjco.jar and sapjcorfc.dll files to the lib directory. Ensure that you added the librfc32.dll file to the Windows system32 folder.
Cannot connect to your SAP system through Application Explorer. The following error message appears: `Problem activating adapter. (com.ibi.sapr3.SapAdapterException : com.sap.mw.jco.JCO$Exception: (102) RFC_ERROR_COMMUNICATION: Connect to SAP gateway failed Connect_PM GWHOST=isdsrv8, GWSERV=sapgw00, ASHOST=isdsrv8, SYSNR=00 LOCATION CPIC (TCP/IP) on local host ERROR partner not reached (host isdsrv8, service 3300) TIME Fri Aug 27 11:49:14 2004 RELEASE 620 COMPONENT NI (network interface) VERSION 36 RC -10 MODULE ninti.c LINE 979 DETAIL NiPConnect2 SYSTEM CALL SO_ERROR ERRNO 10061 ERRNO TEXT WSAECONNREFUSED: Connection refused COUNTER 1). Check logs for more information.`	Ensure that SAP is running and that the parameter values for connecting to your server are correct.

Error	Solution
Cannot connect to your SAP system through Application Explorer even though SAP is running. The following error message appears: `Problem activating adapter. (com.ibi.sapr3.SapAdapterException : java.lang.ExceptionInInitializerError: JCO.classInitialize(): Could not load middleware layer 'com.sap.mw.jco.rfc.MiddlewareRFC' JCO.nativeInit(): Could not initialize dynamic link library sapjcorfc [no sapjcorfc in java.library.path]. java.library.path`	Ensure that you added the sapjcorfc.dll file to the lib directory and the librfc32.dll file to the Windows system32 folder.

The DLL is loaded in another class loader (iBSE and JCA are installed on the same server). The following error message appears:
`com.ibi.sapr3.SapAdapterException: java.lang.ExceptionInInitializerError: JCO.classInitialize(): Could not load middleware layer 'com.sap.mw.jco.rfc.MiddlewareRFC'`
`JCO.nativeInit(): Could not initialize dynamic link library sapjcorfc [Native Library F:\iWay55.008.0628\lib\sapjcorfc.dll already loaded in another classloader]. java.library.path`
	Ensure that you added the sapjco.jar file to the server class path.

Error Messages in SAP

The following table lists errors and solutions when using the adapter.

Error	Solution
When executing a request, the following error message appears: `AdapterException: java.lang.Exception: Function module CUSTOMER_GETDETAIL2 does NOT exist.`	Check the syntax of your input XML document and ensure the name of the Remote Function module is correct and is available in SAP.
When executing a request, the following error message appears: `AdapterException: java.lang.Exception: Object type unknown for business object: CUST`	Check the syntax of your input XML document and ensure the Object type is correct.
When executing a request, the following error message appears: `AdapterException: java.lang.Exception: Unable to retrieve BAPI name for: CUSTOMER.DETAIL2`	Check the syntax of your input XML document and ensure the name of the BAPI is correct and is available in SAP.
When executing a request, the following error message appears: `java.lang.RuntimeException: com.sap.mw.jco.JCO$AbapException: (126) OBJECT_UNKNOWN: Basic type or extension does not exist.`	Check the syntax of your input XML document and ensure the IDoc extension is correct and is available in SAP.
When executing a request, the following error message appears: `AdapterException: java.lang.Exception: BapiError/BapiAbort: You are not authorized to display customers.`	Make sure your user ID has the correct permissions configured in SAP. For more information, consult your SAP administrator.

Error Messages in JCA

The following table lists an error and a solution when using 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, *iway_home*`/lib`

Error Messages in iBSE

The following topics discuss the different types of errors that can occur when processing Web services through the Integration Business Services Engine (iBSE).

General Error Handling in iBSE

The Integration Business Services Engine (iBSE) serves as both a SOAP gateway into the adapter framework and as the engine for some of the adapters. At design time and run time, various conditions can cause errors in iBSE when Web services that use adapters are running. Some 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 to expect when you encounter 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, the way 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, when 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 the previous 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 run time, 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. Because adapters use the target system interfaces and APIs, whether 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.

Although it is almost impossible to anticipate every error condition that an adapter may encounter, the following examples show how adapters handle common error conditions and how error conditions are then exposed to the Web services consumer application.

Example: BEA WebLogic Adapter for SAPInvalid SOAP Request

When the BEA WebLogic Adapter for SAP receives a SOAP request message that does not conform to the WSDL for the Web service 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>Error processing agent [XDSapIfrAgent] - XD[FAIL]
       SapIFRException: java.sql.SQLException:
       com.ibi.sapjco.SapCallableStatement: execute() j
       java.util.NoSuchElementException</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example: Empty Result From SOAP Request

When the BEA WebLogic Adapter for SAP executes an SAP object as a Web service using input parameters passed in the SOAP request message that do not match records in SAP, 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>Error processing agent [XDSapIfrAgent] - XD[FAIL]
     SapIFRException: java.sql.SQLException:
     com.ibi.sapjco.SapCallableStatement: execute()
     java.sql.SQLException: JCO Error Key: NO_RECORD_FOUND Short
     Description: com.sap.mw.jco.JCO$AbapException: (126)
     NO_RECORD_FOUND: NO_RECORD_FOUND</faultstring>
   </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example: Failure to Connect to SAP

When the BEA WebLogic Adapter for SAP cannot connect to SAP when executing a Web service, 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>Error processing agent [XDSapIfrAgent] - XD[RETRY]
       Connect to SAP gateway failed Connect_PM GWHOST=ESDSUN,
       GWSERV=sapgw00, ASHOST=ESDSUN, SYSNR=00 LOCATION CPIC (TCP/IP) on
       local host ERROR partner not reached (host ESDSUN, service 3300)
       TIME Mon Jun 30 16:01:02 2003 RELEASE 620 COMPONENT NI (network
       interface) VERSION 36 RC -10 MODULE ninti.c LINE 976 DETAIL
       NiPConnect2 SYSTEM CALL SO_ERROR ERRNO 10061 ERRNO TEXT
       WSAECONNREFUSED: Connection refused COUNTER 1</faultstring>
    </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example: Invalid SOAP Request

When the BEA WebLogic Adapter for SAP receives a SOAP request message that does not conform to the WSDL for the Web 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>
```

Example: Empty Result From an BEA WebLogic Adapter for SAP SOAP Request

Note: The condition for this adapter does not yield a SOAP fault.

When the BEA WebLogic Adapter for SAP executes a SOAP request using input parameters passed that do not match records in the target system, 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: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

Sample Files and Coding Techniques

Topics:

- Sample RFC Request Document
- Sample RFC Response Document
- Sample IDoc XML for Message Type DEBMAS
- Sample RFC Module
- Sample Wrapper Module to Call Functions on Remote Destinations (Service)
- Using Staging BAPIs to Retrieve SAP BW Metadata

This section provides sample request and response documents sent between SAP and the BEA WebLogic Adapter for SAP. It also includes a sample RFC module and a sample wrapper module to call functions on remote destinations.

Sample RFC Request Document

The following sample code shows a sample RFC request document.

```
<?xml version="1.0" ?>
<doc:RFC_WALK_THRU_TEST xmlns:doc="urn:sapcom:document:sap:business:rfc">
   <TEST_IN>
      <RFCFLOAT>0.0</RFCFLOAT>
      <RFCCHAR1></RFCCHAR1>
      <RFCINT2>0</RFCINT2>
      <RFCINT1>0</RFCINT1>
      <RFCCHAR4></RFCCHAR4>
      <RFCINT4>10</RFCINT4>
      <RFCHEX3>000000</RFCHEX3>
      <RFCCHAR2></RFCCHAR2>
      <RFCTIME>10:09:32</RFCTIME>
      <RFCDATE>2001-09-05</RFCDATE>
      <RFCDATA1>Hello World</RFCDATA1>
      <RFCDATA2></RFCDATA2>
   </TEST_IN>
   <DESTINATIONS>
   </DESTINATIONS>
   <LOG>
   </LOG>
</doc:RFC_WALK_THRU_TEST>
```

Sample RFC Response Document

The following sample code shows a sample RFC response document.

```
<?xml version="1.0" ?>
<doc:RFC_WALK_THRU_TEST.Response
 xmlns:doc="urn:sapcom:document:sap:business:rfc">
   <TEST_OUT>
      <RFCFLOAT>0.0</RFCFLOAT>
      <RFCCHAR1></RFCCHAR1>
      <RFCINT2>0</RFCINT2>
      <RFCINT1>0</RFCINT1>
      <RFCCHAR4></RFCCHAR4>
      <RFCINT4>10</RFCINT4>
      <RFCHEX3>000000</RFCHEX3>
      <RFCCHAR2></RFCCHAR2>
      <RFCTIME>10:09:32</RFCTIME>
      <RFCDATE>2001-09-05</RFCDATE>
      <RFCDATA1>Hello World</RFCDATA1>
      <RFCDATA2></RFCDATA2>
   </TEST_OUT>
   <DESTINATIONS>
   </DESTINATIONS>
   <LOG>
   </LOG>
</doc:RFC_WALK_THRU_TEST.Response>
```

Sample IDoc XML for Message Type DEBMAS

The following sample code shows a sample IDoc XML document for the DEBMAS message type.

```
<?xml version="1.0" ?>
<DEBMAS01>
   <IDOC BEGIN="1">
      <EDI_DC40 SEGMENT="1">
         <TABNAM>EDI_DC40</TABNAM>
         <MANDT>800</MANDT>
<DOCNUM>0000000000236015</DOCNUM>
         <DOCREL>46C</DOCREL>
         <STATUS>30</STATUS>
         <DIRECT>1</DIRECT>
         <OUTMOD>2</OUTMOD>
         <EXPRSS></EXPRSS>
         <TEST></TEST>

         <IDOCTYP>DEBMAS01</IDOCTYP>
         <CIMTYP></CIMTYP>
         <MESTYP>DEBMAS</MESTYP>
         <MESCOD></MESCOD>
         <MESFCT></MESFCT>
         <STD></STD>

         <STDVRS></STDVRS>
         <STDMES></STDMES>
         <SNDPOR>SAPI46</SNDPOR>
         <SNDPRT>LS</SNDPRT>
         <SNDPFC></SNDPFC>
         <SNDPRN>I46_CLI800</SNDPRN>
         <SNDSAD></SNDSAD>
         <SNDLAD></SNDLAD>

         <RCVPOR>A000000018</RCVPOR>
         <RCVPRT>LS</RCVPRT>
         <RCVPFC></RCVPFC>
         <RCVPRN>SAMP</RCVPRN>
         <RCVSAD></RCVSAD>
         <RCVLAD></RCVLAD>
         <CREDAT>2001-09-04</CREDAT>
         <CRETIM>16:44:52</CRETIM>
         <REFINT></REFINT>
         <REFGRP></REFGRP>
         <REFMES></REFMES>
         <ARCKEY></ARCKEY>
         <SERIAL>20010904164452</SERIAL>
      </EDI_DC40>
```

```
<E1KNA1M SEGMENT="1">
   <MSGFN>005</MSGFN>
   <KUNNR>0000000001</KUNNR>
   <ANRED></ANRED>
   <AUFSD></AUFSD>
   <BAHNE></BAHNE>
   <BAHNS></BAHNS>
   <BBBNR>0000000</BBBNR>
   <BBSNR>00000</BBSNR>

   <BEGRU></BEGRU>
   <BRSCH></BRSCH>
   <BUBKZ>0</BUBKZ>
   <DATLT></DATLT>
   <FAKSD></FAKSD>
   <FISKN></FISKN>
   <KNRZA></KNRZA>
   <KONZS></KONZS>

   <KTOKD>0001</KTOKD>
   <KUKLA></KUKLA>

   <LAND1>US</LAND1>
   <LIFNR></LIFNR>
   <LIFSD></LIFSD>
   <LOCCO></LOCCO>
   <LOEVM></LOEVM>
   <NAME1>Apple Corp</NAME1>
   <NAME2></NAME2>
   <NAME3></NAME3>
   <NAME4></NAME4>

   <NIELS></NIELS>
   <ORT01>Floral Park</ORT01>
   <ORT02></ORT02>
   <PFACH></PFACH>
   <PSTL2></PSTL2>
   <PSTLZ>10010</PSTLZ>
   <REGIO>NY</REGIO>
   <COUNC></COUNC>
   <CITYC></CITYC>
   <RPMKR></RPMKR>
   <SORTL>APPLE</SORTL>
   <SPERR></SPERR>
   <SPRAS>E</SPRAS>
   <STCD1></STCD1>
   <STCD2></STCD2>
   <STKZA></STKZA>
   <STKZU></STKZU>
   <STRAS>123 Main street</STRAS>
```

```
<TELBX></TELBX>
<TELF1></TELF1>
<TELF2></TELF2>
<TELFX></TELFX>
<TELTX></TELTX>
<TELX1></TELX1>
<LZONE>0000000001</LZONE>

<XZEMP></XZEMP>
<VBUND></VBUND>
<STCEG></STCEG>
<GFORM></GFORM>
<BRAN1></BRAN1>
<BRAN2></BRAN2>
<BRAN3></BRAN3>
<BRAN4></BRAN4>
<BRAN5></BRAN5>
<UMJAH>0000</UMJAH>

<UWAER></UWAER>
<JMZAH>000000</JMZAH>
<JMJAH>0000</JMJAH>
<KATR1></KATR1>
<KATR2></KATR2>
<KATR3></KATR3>
<KATR4></KATR4>
<KATR5></KATR5>
<KATR6></KATR6>
<KATR7></KATR7>
<KATR8></KATR8>
<KATR9></KATR9>
<KATR10></KATR10>
<STKZN></STKZN>
```

```
<UMSA1>0</UMSA1>
<TXJCD></TXJCD>
<PERIV></PERIV>
<KTOCD></KTOCD>
<PFORT></PFORT>
<DTAMS></DTAMS>
<DTAWS></DTAWS>
<HZUOR>00</HZUOR>
<CIVVE>X</CIVVE>
<MILVE></MILVE>
<SPRAS_ISO>EN</SPRAS_ISO>
<FITYP></FITYP>
<STCDT></STCDT>
<STCD3></STCD3>
<STCD4></STCD4>
<XICMS></XICMS>
<CFOPC></CFOPC>

<TXLW1></TXLW1>
<TXLW2></TXLW2>
<CCC01></CCC01>
<CCC02></CCC02>
<CCC03></CCC03>
<CCC04></CCC04>
<CASSD></CASSD>
<KDKG1></KDKG1>
<KDKG2></KDKG2>
<KDKG3></KDKG3>
<KDKG4></KDKG4>
<KDKG5></KDKG5>
<NODEL></NODEL>
<XSUB2></XSUB2>
<WERKS></WERKS>
```

```
<E1KNVVM SEGMENT="1">
    <MSGFN>005</MSGFN>
    <VKORG>0001</VKORG>
    <VTWEG>01</VTWEG>
    <SPART>01</SPART>
   <BEGRU></BEGRU>
   <LOEVM></LOEVM>
   <VERSG></VERSG>
   <AUFSD></AUFSD>
   <KALKS>1</KALKS>
   <KDGRP></KDGRP>
   <BZIRK></BZIRK>
   <KONDA></KONDA>
   <PLTYP></PLTYP>
   <AWAHR>100</AWAHR>
   <INCO1></INCO1>
   <INCO2></INCO2>
   <LIFSD></LIFSD>
   <AUTLF></AUTLF>
   <ANTLF>9</ANTLF>
   <KZTLF></KZTLF>
   <KZAZU>X</KZAZU>
   <CHSPL></CHSPL>
   <LPRIO>00</LPRIO>
   <EIKTO></EIKTO>
   <VSBED>01</VSBED>
   <FAKSD></FAKSD>
   <MRNKZ></MRNKZ>
   <PERFK></PERFK>
   <PERRL></PERRL>
   <WAERS>EUR</WAERS>
   <KTGRD></KTGRD>

    <ZTERM></ZTERM>
    <VWERK></VWERK>
    <VKGRP></VKGRP>
    <VKBUR></VKBUR>
    <VSORT></VSORT>
    <KVGR1></KVGR1>
    <KVGR2></KVGR2>
    <KVGR3></KVGR3>
    <KVGR4></KVGR4>
    <KVGR5></KVGR5>
```

```
<BOKRE></BOKRE>
<KURST></KURST>
<PRFRE></PRFRE>
<KLABC></KLABC>
<KABSS></KABSS>
<KKBER></KKBER>
<CASSD></CASSD>
<RDOFF></RDOFF>
<AGREL></AGREL>
<MEGRU></MEGRU>
<UEBTO>0.0</UEBTO>
<UNTTO>0.0</UNTTO>
<UEBTK></UEBTK>
<PVKSM></PVKSM>
<PODKZ></PODKZ>
<PODTG>          0</PODTG>

<E1KNVPM SEGMENT="1">
    <MSGFN>005</MSGFN>
    <PARVW>AG</PARVW>
    <KUNN2>0000000001</KUNN2>
    <DEFPA></DEFPA>
    <KNREF></KNREF>
    <PARZA>000</PARZA>
</E1KNVPM>

<E1KNVPM SEGMENT="1">
    <MSGFN>005</MSGFN>
    <PARVW>RE</PARVW>
    <KUNN2>0000000001</KUNN2>
    <DEFPA></DEFPA>
    <KNREF></KNREF>
    <PARZA>000</PARZA>
</E1KNVPM>

<E1KNVPM SEGMENT="1">
    <MSGFN>005</MSGFN>
    <PARVW>RG</PARVW>
    <KUNN2>0000000001</KUNN2>
    <DEFPA></DEFPA>
    <KNREF></KNREF>
    <PARZA>000</PARZA>
</E1KNVPM>
```

```
        <E1KNVPM SEGMENT="1">
            <MSGFN>005</MSGFN>
            <PARVW>WE</PARVW>
            <KUNN2>0000000001</KUNN2>
            <DEFPA></DEFPA>
            <KNREF></KNREF>
            <PARZA>000</PARZA>
        </E1KNVPM>

        <E1KNVIM SEGMENT="1">
             <MSGFN>005</MSGFN>
             <ALAND>DE</ALAND>
             <TATYP>MWST</TATYP>
             <TAXKD>0</TAXKD>
        </E1KNVIM>

    </E1KNVVM>
  </E1KNA1M>
 </IDOC>
</DEBMAS01
```

Collected IDocs

When using collected IDocs on any platform during inbound processing (service mode), if the DOCNUM field does not have a unique document number for each IDoc, the system creates an IDoc for each header record in the collected IDoc file and duplicates the data for each IDoc.

Make sure the DOCNUM field is included in the EDI_DC40 structure and that each IDoc has a unique sequence number within the collected IDoc file.

Sample RFC Module

After you have configured the SAP event adapter and the RFC destination, you can write ABAP code to execute calls at your new destination (the event adapter).

The following sample code uses a user-defined RFC module named Z_EVENT_DISPATCH.

```
FUNCTION Z_01_EVENT_DISPATCH.
CALL FUNCTION 'Z_EVENT_DISPATCH'
  DESTINATION 'IWAYDEST'
  EXPORTING
    EVENT = EVENT
    RECTYPE = RECTYPE
    OBJTYPE = OBJTYPE
    OBJKEY = OBJKEY
  TABLES
    EVENT_CONTAINER = EVENT_CONTAINER.
ENDFUNCTION.
```

Sample Wrapper Module to Call Functions on Remote Destinations (Service)

This topic describes how to invoke a service that employs SAP remote data. For example, you can use this technique to write a function using C on a UNIX server that queries an Informix database and returns the response to SAP.

The ABAP command, CALL FUNCTION, takes as an argument, DESTINATION. Using RFC (Remote Function Call) destinations, programs can be executed on external systems, and the results can be returned into SAP function module programs. For more information on this functionality, see your SAP documentation, which is available at the following URL:

`http://help.sap.com`

Since DESTINATION is not part of an individual BAPI (Business Application Programming Interface) or Remote Function Module (RFM), but a parameter of the SAP function mechanism, you require a *wrapper module* to invoke it as a service. In addition, you must invoke the wrapper module in place of the original function.

The wrapper module is written using SAP's ABAP/4 programming language and contains the same input and output parameters as the original function. You can obtain all of the parameters of a remote function in the function editor by selecting Edit and then, Pattern and entering the function name.

The destination inside the wrapper module must be a valid SAP RFC destination with an RFC Server program running on the remote host. For more information, see SAP's RFC Programming manual, which is available at the following URL:

`http://service.sap.com`

The RFC Server program must return the data to SAP in a format that follows the exact structure of the Remote Function interface, or an abnormal ending occurs in SAP.

The following is an example of a wrapper module for the SAP test function named RFC_CUSTOMER_GET.

```
FUNCTION Z_CALL_EXTERNAL.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  IMPORTING
*"     VALUE(MYKUNNR) LIKE  KNA1-KUNNR DEFAULT SPACE
*"     VALUE(MYNAME1) LIKE  KNA1-NAME1 DEFAULT SPACE
*"  EXPORTING
*"     VALUE(ERRORCODE) LIKE  SY-SUBRC
*"  TABLES
*"      MYCUSTOMER_T STRUCTURE  BRFCKNA1
*"----------------------------------------------------------------------
ERRORCODE = 0.
CALL FUNCTION 'RFC_CUSTOMER_GET'
 DESTINATION 'JRDEST'
  EXPORTING
    KUNNR                    = MYKUNNR
    NAME1                    = MYNAME1
    TABLES
    CUSTOMER_T               = MYCUSTOMER_T
EXCEPTIONS
      COMMUNICATION_FAILURE = -1
      SYSTEM_FAILURE        = -2
      NOTHING_SPECIFIED     = -3
      NO_RECORD_FOUND       = -4
      OTHERS                = -5.
CASE SY-SUBRC.
     WHEN 0.
     ERRORCODE = 0.
     EXIT.
     WHEN -1 .
      ERRORCODE = 1.
     EXIT.
     WHEN -2.
      ERRORCODE = 2.
     EXIT.
     WHEN -3.
      ERRORCODE   =  3.
      EXIT.
      WHEN -4.
      ERRORCODE = 4.
      EXIT.
     WHEN -5.
     ERRORCODE  = 99999.
      EXIT.
  ENDCASE.
          .
* IF SY-SUBRC <> 0.
*ERRORCODE = SY-SUBRC.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
*ENDIF.
ENDFUNCTION.
```

Using Staging BAPIs to Retrieve SAP BW Metadata

The Staging (or Warehouse Management) BAPIs (Business Application Programming Interfaces) include methods to update and retrieve metadata for InfoObjects, InfoCubes, InfoObjectCatalogs, and the definition of InfoPackages, an SAP Business Warehouse (BW), from a third party tool.

By using these BAPIs, you can connect metadata repositories and extraction engines to the SAP Business Information Warehouse. BEA WebLogic enables you to link these systems to the rest of the enterprise.

For complete documentation on the individual data structures and individual BAPI calls, see Business Information Warehouse available at the following URL:

`http://service.sap.com`

Procedure: How to Deploy BAPIs Through BEA WebLogic

To deploy BAPIs through BEA WebLogic:

1. Start the Servlet Application Explorer and connect to SAP.
2. Select an object to query.

   In this procedure, you use BAPI_INFOCUBE_GETLIST as an example.
3. Create a schema.
4. Create an XML instance from the schema.
5. Modify the XML by entering *A* for active version in the version tag as follows:

   ```
   <VERSION>A</VERSION>

   </InfoObject.GetList>
   ```

   The goal is to retrieve a list of all active InfoCubes, so the character, A, is entered in the VERSION parameter of the tag.

6. Start BEA WebLogic and configure a File event.

When the XML document is placed into the specified file listener directory, BEA WebLogic automatically submits it to SAP for processing and returns the document with a list of all active InfoCubes from the SAP Business Warehouse.

For example, the following is the request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<InfoCube.GetList.Response xmlns="urn:sap-com:document:sap:business"
 schemaLocation="urn:sap-com:document:sap:business
 C:\PROGRA~1\COMMON~1\iway\Adapters\5.2.104\sessions\default\
 sap\GAH\service_BAPI_CUBE_GETLIST_response.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <RETURN>
   <TYPE/>
   <ID/>
   <NUMBER>000</NUMBER>
   <MESSAGE/><LOG_NO/><LOG_MSG_NO>000000</LOG_MSG_NO>
   <MESSAGE_V1/><MESSAGE_V2/><MESSAGE_V3/>
   <MESSAGE_V4/><PARAMETER/><ROW>0</ROW><FIELD/><SYSTEM/>
  </RETURN>
```

For example, the following is the response:

```
<INFOCUBELIST><item><INFOCUBE>0BWTCFC1</INFOCUBE><OBJVERS>A</OBJVERS>
<TEXTLONG>BW Technical Content FC1</TEXTLONG><OBJSTAT>ACT</OBJSTAT>
<ACTIVFL>X</ACTIVFL><INFOAREA>0BWTCT_FCHA</INFOAREA>
<CUBETYPE>B</CUBETYPE>      </item><item><INFOCUBE>0BWTCFC2</INFOCUBE>
<OBJVERS>A</OBJVERS><TEXTLONG>BW Technical Content 2</TEXTLONG>
<OBJSTAT>ACT</OBJSTAT><ACTIVFL>X</ACTIVFL>
<INFOAREA>0BWTCT_FCHA</INFOAREA><CUBETYPE>B</CUBETYPE>
      </item>
.
.
.
</INFOCUBELIST></InfoCube.GetList.Response>
```

APPENDIX B

Best Practices and Performance Considerations

Topics:

- Overview of the BEA WebLogic Adapter for SAP
- Supported Tools and Technologies
- Understanding Web Services and Java Connector Architecture Functionality
- Understanding JCA Managed and Non-managed Modes
- Application Design
- Configuration Tuning

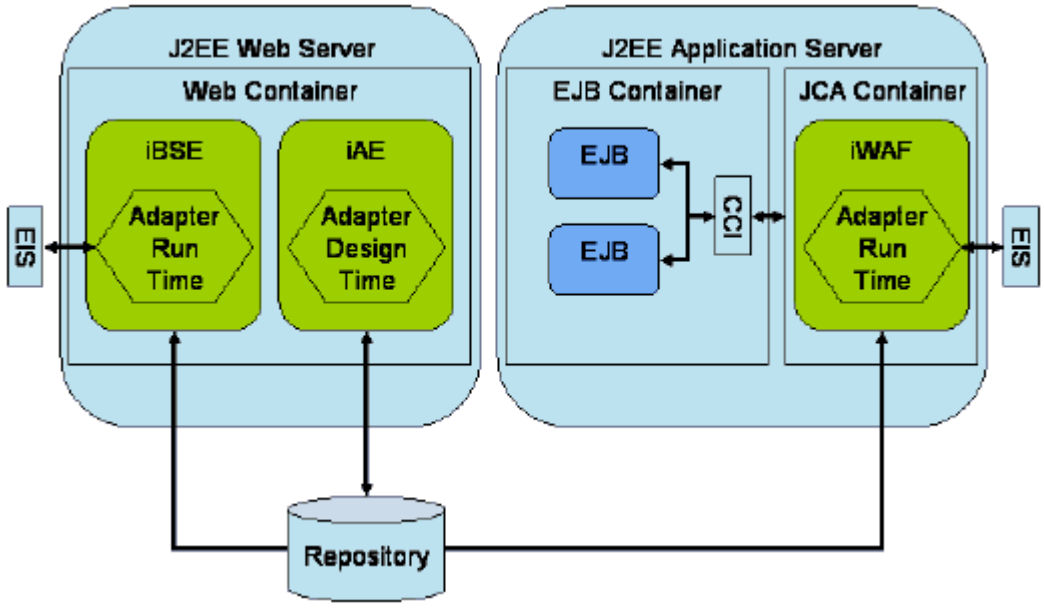
This topic provides best practices and performance considerations for the BEA WebLogic Adapter for SAP when deployed to the BEA WebLogic Server.

Overview of the BEA WebLogic Adapter for SAP

The BEA WebLogic Adapter for SAP provides a means to exchange real-time business data between SAP systems and other application, database, or external business partner systems. It supports asynchronous, bi-directional interactions between the adapter and your SAP R/3 applications, and enables business process workflows running within BEA WebLogic to transfer data to and from an SAP R/3 application. The adapter allows you to execute SAP IFR XML, IDocs, BAPI calls, or custom RFCs.

Adapter Framework

The following diagram illustrates the BEA WebLogic Adapter for SAP framework.



The Integration Business Services Engine (iBSE) and Application Explorer (iAE) are deployed to the J2EE Web container. The Connector for JCA is deployed to the JCA container in the J2EE server. The repository is written at design time and read at run time. The container (iAE, iWAF, or iBSE) is responsible for the connection to the repository as shown.

Supported Versions and Platforms

The BEA WebLogic Adapter for SAP supports R/3 and R/3 Enterprise. The following SAP R/3 versions are supported by the BEA WebLogic Adapter for SAP:

- 4.6C
- 4.6D

- SAP R/3 Enterprise 47x100

  SAP R/3 Enterprise Version 47x100 is supported on the SAP Web Application Server Versions 6.20 and 6.40.

- SAP R/3 Enterprise 47x200

  SAP R/3 Enterprise Version 47x200 is supported on the SAP Web Application Server Versions 6.20 and 6.40.

- SAP Java Connector (SAP JCo) 2.16

  Note: For the current release status of the SAP Java Connector, refer to SAP Note #549268 in the SAP Service Marketplace.

In addition, the BEA WebLogic Adapter for SAP supports those SAP versions that are on the mainstream maintenance track as defined by the SAP Service Marketplace release strategy. To access this document, use the following URL:

`http://service.sap.com/releasestrategy/`

If the SAP R/3 version you are looking for is not listed in this table, please consult your BEA Customer Service Representative.

The following 32-bit platforms are supported by the BEA WebLogic Adapter for SAP:

- Windows 2000/XP/2003
- Linux (Intel compatible) 32-bit
- Solaris (SPARC processors only)
- AIX
- HP-UX
- TRU64
- OS/400
- Z/OS

Integrating With SAP

You can use the BEA WebLogic Adapter for SAP to invoke an SAP business process, such as add/update account, or you can use the adapter as part of an integration effort to connect SAP and non-SAP systems.

BAPIs and RFCs are called synchronously by the adapter and always return data (either technical error information or a well-formed response document). IDocs are processed asynchronously.

The adapter is bidirectional and can process an event in SAP by receiving RFCs and IDocs directly from SAP. The SAP system can be configured to send an IDoc or RFC out to a logical system when a certain event occurs, in this case to the adapter. The output sent by SAP can be in any of the following forms:

- An RFC request, for example, RFC_SYSTEM_INFO.
- A BAPI request, for example, BAPI_COMPANYCODE_GETLIST.
- An IDoc.

For request processing, the BEA WebLogic Adapter for SAP can send requests to SAP using the BAPI, RFC, or IDoc interfaces.

The adapter quickly and easily integrates your SAP IDocs, RFCs, and BAPIs with mission critical SAP system applications and other enterprise applications. Adapter benefits include:

- Eliminating the requirement for custom coding.
- Consistent data representation—a standard XML representation of event data and request/response documents for SAP. The developer is freed from the specific details of the SAP interface (BAPI, RFC, IDoc,) and the specific configuration details of the target SAP system.
- Adherence to SAP ABAP serialization rules and SAP Interface Repository standards published by SAP AG.

Supported Tools and Technologies

The BEA WebLogic Adapter for SAP works in conjunction with the following components:

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

Application Explorer

Application Explorer uses an explorer metaphor to browse the SAP system for metadata. The explorer enables you to create XML schemas and Web services for the associated object. In addition, you can create ports and channels to listen for events in your SAP application. External applications that access SAP through the BEA WebLogic Adapter for SAP use either XML schemas or Web services to pass data between the external application and the adapter.

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, 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.

Enterprise Connector for J2EE Connector Architecture (JCA)

The Enterprise Connector for J2EE Connector Architecture (JCA) enables developers of JCA-compliant applications to deploy adapters as JCA resources. The connector is supported on J2EE-compliant application servers, such as your application server.

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.

Understanding Web Services and Java Connector Architecture Functionality

The following section describes how the BEA WebLogic Adapter for SAP can incorporate Web services and Java Connector Architecture technology.

Web Services

Web services allow SAP calls to be made across the Internet or an intranet, using specialized versions of the XML language that allow a developer to specify the parameters, connections methods, and remote calls and store them for reference in a repository. At runtime, a person, an interface, or another function, can read this repository and automatically invoke the service. Web services currently do not have industry standards for transactional behavior. Web services are useful when your function calls must be made across firewall boundaries. Using Web services, you can use functions provided by external providers, as long as you know the function interface.

Java Connector Architecture

Java Connector Architecture (JCA) provides a reusable component model to build and deploy multi-tier applications that are platform and vendor-independent. JCA acts as a type of envelope or "container" that will allow the adapter to run inside the application server and connect to SAP and immediately return the results. JCA is useful when your SAP system resides within a local intranet or is accesed directly. JCA implements JAVA Connection and Transaction models. JCA requires a resource adapter to be physically deployed on the host application server to access the remote EIS system.

Using combinations of JCA and Web services is possible. For example, a JCA application can be invoked via a Web service or a Web service may be implemented inside a JCA container. The standards and protocols are still evolving.

Understanding JCA Managed and Non-managed Modes

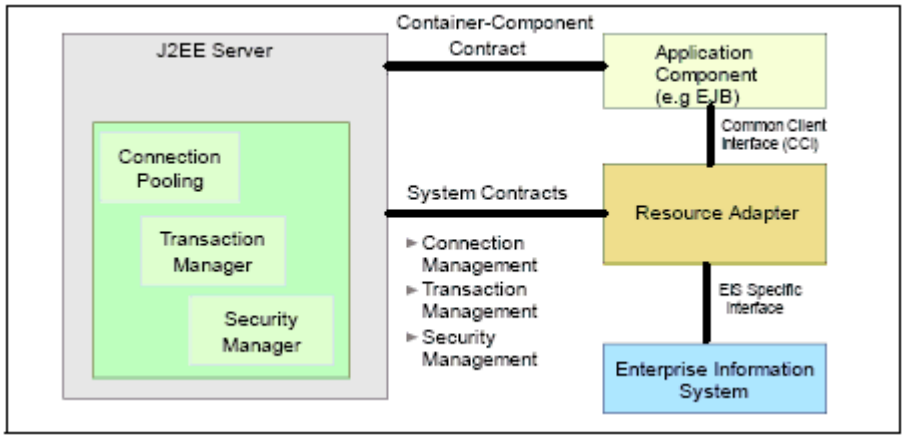
The J2EE Connector Architecture (JCA) provides a specification that standardizes access to Enterprise Information Systems (EIS). When a connection to an EIS is made, this connection can exist in a managed or non-managed mode, depending on your environment and deployment method.

The following section provides an overview of JCA managed and non-managed modes and their key differences. In addition, information relating to the deployment of the Connector for JCA in a managed or non-managed mode is provided.

JCA Managed Mode

By default, the Connector for JCA runs in managed mode when deployed to an application server, for example, BEA WebLogic. In this case, the BEA WebLogic server manages the connections, transactions, and security. As a result, the Connector for JCA provides features such as connection pooling. Connection pooling allows the BEA WebLogic server to pool connections to a back-end EIS, which results in efficient performance and increased scalability.

The following diagram illustrates the Connector for JCA (Resource Adapter) being used in managed mode. In this scenario, the application is running inside the J2EE application server.



In order to make connections to an EIS in managed mode, the Connector for JCA obtains an `IWAFConnectionFactory` from the BEA WebLogic server. In managed mode, the `IWAFConnectionFactory` is accessed from the JNDI and its properties will be configured by the BEA WebLogic server.

The following connection properties can be found in the weblogic-ra.xml file, which is packaged in the iwafjca.rar file.

```
<connection-instance>
        <jndi-name>eis/IWAFConnectionFactory</jndi-name>
         <connection-properties>
            <pool-params>
               <initial-capacity>0</initial-capacity>
               <max-capacity>10</max-capacity>
            </pool-params>
         </connection-properties>
</connection-instance>
```

The following connection properties can be found in the web.xml file, which is used by the JCA IVP test tool to run in managed mode.

```
<context-param>
    <description>JNDI name for the IWAF JCA Resource Adapter.
      If not provided, the application will create a new one based on
      iway.home, iway.config and iway.loglevel.
    </description>
    <param-name>iway.jndi</param-name>
    <param-value>eis/IWAFConnectionFactory</param-value>
</context-param>
```

The iway.jndi parameter is the connection factory name for the Connector for JCA. The connection factory name under BEA WebLogic is eis/IWAFConnectionFactory. The JCA IVP test tool attempts to connect to the adapter via JNDI if JNDI is defined (managed mode). If JNDI is undefined, iway.home, iway.config, and iway.loglevel are used instead (non-managed-mode).

The default location of the web.xml file on Windows is:

```
C:\Program Files\iWay55\bea\iwfjcaivp\WEB-INF\web.xml
```

JCA Non-managed Mode

You can use the Connector for JCA in non-managed mode by embedding it in an application instead of deploying it to an application server. In this case, the application must manage connections, transactions, and security itself. To enable non-managed mode for the Connector for JCA, you must specify the iWay55\lib connector library in the classpath. No RAR file or ra.xml descriptor is needed. In non-managed mode, the default connection manager is used.

Performance may be reduced when deploying the Connector for JCA in non-managed mode since connection pooling is not available.

Application Design

This section addresses best practice principles that can be used during application design stages when using the BEA WebLogic Adapter for SAP.

Web Services and JCA Usage Considerations

The following four factors explain the differences between deploying Web services and deploying the JCA option. Understanding the factors can help in selecting a deployment option.

1. Web services is the preferred deployment option because JCA:

- Can be deployed in a separate instance of BEA WebLogic server.

- Provides better distribution of load.
- Provides better isolation from any errors from third-party libraries.
- Provides better capability to isolate issues for debugging purposes.
- Conforms more closely to SOA model for building applications.

2. JCA provides slightly better performance

JCA does provide slightly better performance than the Web services option; however, the difference decreases as the transaction rate increases.

3. The Web services and JCA options both provide identity propagation at run time.

The Web services option provides the capability to pass identity using the SOAP header. For the JCA option, user name and password can be passed using the connection spec of the CCI.

4. Transactions

Because no adapters currently being resold by BEA support XA transactions, transactions are not a consideration when choosing between JCA and Web services.

Configuring a Clustered Environment

Events can be configured in a clustered BEA WebLogic environment. You can deploy iBSE or JCA to this 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.

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.

A WebLogic cluster is a group of servers that, from the application point-of-view, operate as a single server. A cluster provides:

- **Scalability.** Additional servers can be added to the cluster dynamically to increase capacity.
- **High-availability.** Redundancy of multiple servers insulates applications from failures.

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

For more information on configuring WebLogic Integration for deployment in a clustered environment, see the *Deploying WebLogic Integration Solutions* documentation.

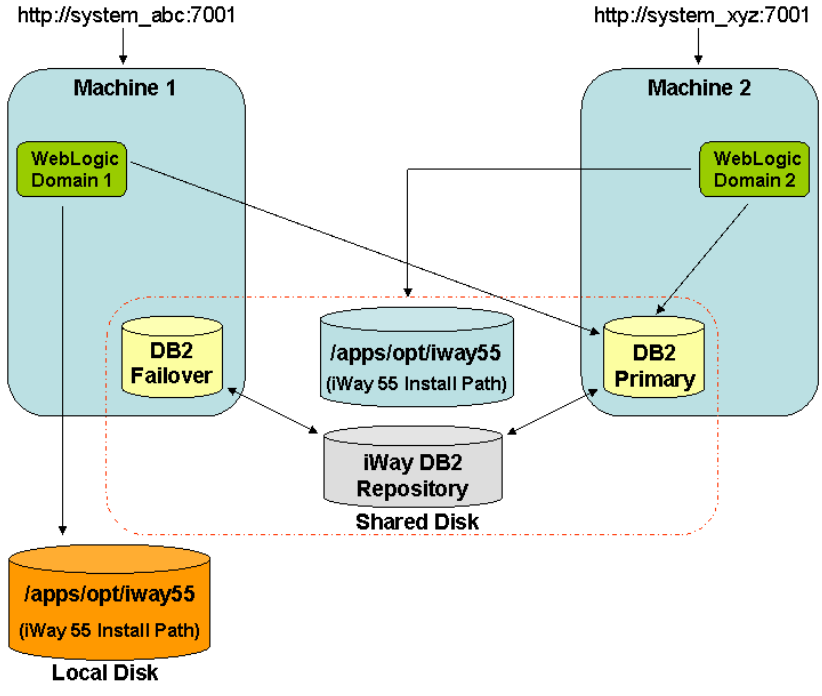
Load Balancing Support with WebLogic Server

The WebLogic Administration Console provides you with the tools to configure WebLogic for load balancing. Load balancing is one of the primary tools employed in making systems scalable. BEA WebLogic Adapter for SAP takes advantage of the sophisticated load balancing features of WebLogic clusters.

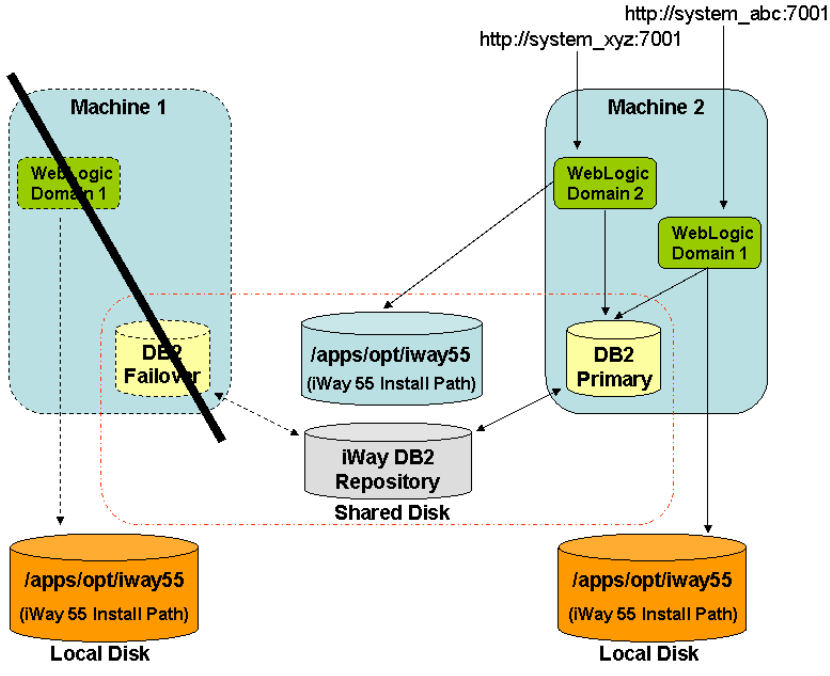
Failover Support with WebLogic Server

Failover is the ability of a system to detect a service failure and automatically retry the operation using another Gateway. This includes the ability to recover in the event of a server failure and the ability to find another instance of a service on an available server. Under the best of conditions the results of a service are idempotent, meaning that multiple invocations of the service, with the same input, produce the same output with no side effects. An example of an idempotent service would be a currency translation service. The service produces the same output each time it is run for a given input value and it has no side effects. Under these conditions it is safe to failover a request for any type of failure.

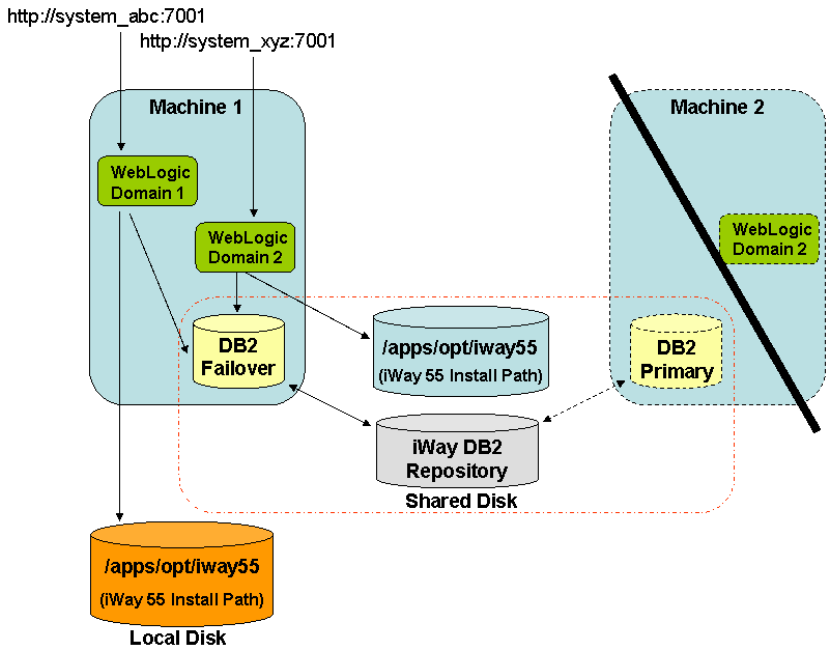
The following diagram illustrates a sample configuration where two WebLogic domains are configured on two separate machines. DB2 Failover and DB2 Primary represent a scenario where DB2 is configured for each WebLogic domain respectively. In the event of a failover condition, the DB2 Primary database instance would be switched over to the DB2 Failover instance.


http://system_abc:7001
http://system_xyz:7001
Machine 1
Machine 2
WebLogic Domain 1
WebLogic Domain 2
DB2 Failover
/apps/opt/iway55
(iWay 55 Install Path)
DB2 Primary
iWay DB2 Repository
Shared Disk
/apps/opt/iway55
(iWay 55 Install Path)
Local Disk

The following diagram illustrates a failover situation when the first machine becomes unavailable. As you can see, Machine 2 takes over in the clustered environment.



The following diagram illustrates a failover situation when the second machine becomes unavailable. In this case, Machine 1 takes over in the clustered environment.



Synchronous Event Processing

You can configure synchronous event processing using Application Explorer to trigger a Web service once an event occurs in the SAP system. The event response that is received can then be routed to another disposition for further processing.

Procedure How to Configure Synchronous Event Processing Using Application Explorer

To configure synchronous event processing:

1. Create a Web service for an SAP Remote Function Module, for example, BAPI_MATERIAL_GETLIST.
2. View the WSDL file.
3. Create a port using the SOAP disposition.

   The Create New Port window opens.

   a. Type a name for the event port and provide a brief description.

   b. From the Disposition Protocol drop-down list, select *SOAP*.

   c. 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 parameters for the disposition.

Parameter	Description

wsdl-url	The URL to the WSDL file that is required to create the SOAP message. For example:
	`http://localhost:7001/ibse/IBSEServlet/test/webservice.ibs?wsdl`
	where:
	`webservice`
	Is the name of the Web service you created using Application Explorer.
	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.

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 can be found in the WSDL file.
namespace	The XML namespace you are using. This value can be found in the WSDL file.
responseTo	The location to which responses are posted, which can be a predefined port name or another URL. Optional. The URL must be complete, including the protocol.
errorTo	The location to which error logs are sent. Optional. A predefined port name or another disposition URL. The URL must be complete, including the protocol.

Note: If you need to use the SOAP disposition with a synchronous event, you should use Remote Function Modules to generate the schema and WSDL file instead of using the Business Object Repository for any RFC or BAPI.

The following is an example of a completed SOAP disposition:

```
soap:http://localhost:7001/ibse/IBSEServlet/test/soapWS.ibs?wsdl;soapaction=soapWS.GETLISTRequest@test@@;method=GETLIST;namespace=urn:iwaysoftware:ibse:jul2003:GETLIST;responseTo=ifile://c:\output\sap\soapOut.xml
```

4. Create a channel.

   The Edit Channels window opens.

5. Provide the required information to connect to SAP in the System and User tabs.
6. In the Advanced tab, select *REQUEST_RESPONSE* from the Processing Mode drop-down list.
7. In the preemitter tab, click the *Strip the SAP Payload* check box.
8. Associate the port you created earlier with the new channel.
9. Start the channel.

   A Web service for an SAP Remote Function Module, for example, BAPI_MATERIAL_GETLIST, is triggered once an event occurs in the SAP system. The response document is returned and routed to a file location.

Configuration Tuning

This section explores tuning considerations during design time and run time.

Repository Migration Using iBSE Administrative Services

iBSE Administrative Services allow a client application to export, import, and modify an existing iBSE repository, regardless of the underlying format. When suitably configured, the iBSE Administrative Services represent a best practice methodology for maintaining separate development, QA, and production environments.

For more information on using iBSE administrative services, see Chapter 4, *Management and Monitoring*.

JCA Troubleshooting

JCA provides the ability to set multiple log levels. The log level is set in the META-INF\ra.xml file.

To change defaults, you must:

1. Extract the META-INF\ra.xml file from the iwafjca.rar archive.

   a. Open a command prompt and navigate to the directory containing the connector, for example:

   ```
   iWay55\etc\setup
   ```

   where:

   ```
   iWay55
   ```

   Is the full path to your iWay 5.5 installation. The default is C:\Program Files\iWay55.

   b. Issue the following command:

   ```
   jar xvf iwafjca.rar META-INF/ra.xml
   ```

   The JAR command is located in the Java SDK bin directory which might not be in your search path. If you receive an error, execute the jar command using its full path. This path varies depending on which version of Java is installed, for example:

   ```
   C:\j2sdk1.4.1_03\bin\jar -xvf iwafjca.rar META-INF/ra.xml
   ```

   Note: Ensure you use the JAR command and not Winzip. Winzip does not properly extract Java related archives.

2. Open the extracted ra.xml file in a text editor.
3. Modify the contents of the <param-value> tags to change defaults.

   - **LogLevel.** Trace setting. This 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>
   ```

   Leave the remainder of this file unchanged.

4. Save the file and exit the editor.

5. Use the JAR command to return the ra.xml file to the META-INF directory within the archive. To do this:

   a. Ensure that you are in the following directory that contains the connector:

      ```
      iWay55\etc\setup
      ```

      where:

      iWay55

      Is the full path of your iWay 5.5 installation directory. The default is C:\Program Files\iWay55.

   b. Issue the following command:

      ```
      jar -uvf iwafjca.rar META-INF/ra.xml
      ```

6. Redeploy the connector.

   The trace information is written to the BEA WebLogic server log file.