WebSphere Adapters

Version 6 Release 2

WebSphere Adapter for SAP Software User Guide
Version 6 Release 2

WebSphere® Adapters

IBM

Version 6 Release 2

WebSphere Adapter for SAP Software User Guide
Version 6 Release 2

Note
Before using this information and the product it supports, read the information in "Notices" on page 257.

December 2008

This edition applies to version 6, release 2, modification 0 of IBM WebSphere Adapter for SAP Software and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email mailto://doc-comments@us.ibm.com. We look forward to hearing from you.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. Overview of WebSphere Adapter for SAP Software 1

Hardware and software requirements 2
Technical overview of WebSphere Adapter for SAP Software 2
- Outbound Processing 5
- Inbound Processing 6
- Adapter Packaging 6
- Business objects 6
- Java data bindings 7
- The J2C Bean wizard 8
- Standards Compliance 8

Chapter 2. Planning for adapter implementation 11

Before you begin 11
Security. 11
Support for protecting sensitive user data in log and trace files 11
User authentication. 12
Deployment options 13
WebSphere Adapters in clustered environments . . 14

Chapter 3. SAP interfaces 17

The BAPI interfaces 17
- Outbound processing for the BAPI interface . . 18
- Inbound processing for the BAPI interface . . . 19
- Java data bindings for the BAPI interface . . . 22
- Business object structure for a simple BAPI. . . 22
- Java data binding structure for a simple BAPI . . 23
- Business object structure for a nested BAPI. . . 24
- Java data binding structure for a nested BAPI . . 24

The BAPI work unit interface 24
- Outbound processing for the BAPI work unit interface 24
- Business object structure for a BAPI work unit . 25
- Java data binding structure for a BAPI work unit 25

The BAPI result set interface. 26
- Outbound processing for the BAPI result set interface 26
- Business object structure for a BAPI result set . . 27
- Java data binding structure for a BAPI result set 27

The ALE interfaces 27
- Outbound processing for the ALE interfaces . . 29
- Inbound processing for the ALE interfaces . . . 29
- Java data bindings for the ALE interface. . . . 35
- Java Data Binding structure for the ALE interface 36

The ALE pass-through IDoc interface. 37
- Outbound processing for the ALE pass-through IDoc interface 38
- Inbound processing for the ALE pass-through IDoc interface 39
- ALE pass-through IDoc business object structure 43

Query interface for SAP Software 43
- Outbound processing for the query interface for SAP Software. 44
- Business objects for the query interface for SAP Software 44

The Advanced event processing interface 45
- Outbound processing for the Advanced event processing interface. 45
- Inbound processing for the Advanced event processing interface. 49
- Business objects for the Advanced event processing interface. 52

Chapter 4. Configuring the module for deployment. 55

Configuration on the SAP system 55
- Configuring the SAP system for ALE or BAPI inbound processing. 55
- Creating the data source 57
- Creating an IDoc definition file. 58
- Adding transport files to the SAP server. . . . 59
- Implementing event-detection mechanisms . . . 59

Launching the J2C Bean wizard 67
Configuring connector dependencies 68
Setting connection properties for the J2C Bean wizard 68
Configuring the module for outbound processing . 70
- Configuring the J2C Bean and Java data bindings for the BAPI interface 70
- Configuring a module for the BAPI work unit interface 77
- Configuring a module for the BAPI result set interface 82
- Configuring the J2C Java bean and Java data bindings for the ALE interface 88
- Configuring a module for ALE pass-through IDoc outbound processing 95
- Configuring a module for Query interface for SAP Software processing 99
- Configuring a module for Advanced event processing - outbound 106

Configuring the module for inbound processing 111
- Configuring a module for BAPI inbound processing 112
- Configuring a module for ALE inbound processing 116
- Configuring a module for ALE pass-through IDoc inbound processing 121
- Configuring a module for Advanced event processing - inbound 123

Chapter 5. Configuring the application on WebSphere Application Server . . 127

Changing configuration properties for embedded adapters 127

Setting resource adapter properties for embedded adapters 127
Setting managed (J2C) connection factory properties for embedded adapters 128
Setting activation specification properties for embedded adapters 129
Changing configuration properties for stand-alone adapters 130
Setting resource adapter properties for stand-alone adapters 130
Setting managed (J2C) connection factory properties for stand-alone adapters 131
Setting activation specification properties for stand-alone adapters 132
Managing Advanced event processing 133
Displaying the current events queue. 133
Displaying the future events queue 134
Maintaining the archive table 135
Managing the adapter log file 136
Monitoring SAP gateway connections 139
Adding dependency libraries to the deployed resource adapter 139
Standalone deployment 139
EAR deployment 140
Using enhanced EAR editor 140
Using administrative console of the WebSphere Application Server 140

Chapter 6. Troubleshooting and support 141
Support for the Log and Trace Analyzer 141
Configuring logging and tracing 141
Configuring logging properties 142
Changing the log and trace file names 143
Detecting errors during outbound processing. . . 144
Resolving errors during Query interface for SAP Software processing 145
Resolving memory-related issues 145
First-failure data capture (FFDC) support 146
XAResourceNotAvailableException 146
Self-help resources. 147

Chapter 7. Reference information . . . 149
Business object information. 149
Application-specific information 149
Supported data operations 156
Naming conventions 159
Outbound configuration properties 163
Connection properties for the wizard 165
Resource adapter properties 174
Managed connection factory properties. . . . 176
Interaction specification properties 187
Inbound configuration properties. 191
Connection properties for the wizard 192
Resource adapter properties 201
Activation specification properties for BAPI inbound processing 204
Activation specification properties for ALE inbound processing 219
Activation specification properties for Advanced event processing 238
Globalization 254
Globalization and bidirectional transformation 254
Properties enabled for bidirectional data transformation 256

Notices 257
Programming interface information 259
Trademarks and service marks 259

Index 261

Chapter 1. Overview of WebSphere Adapter for SAP Software

With WebSphere Adapter for SAP Software, you can create integrated processes that include the exchange of information with an SAP server, without special coding.

Using the adapter, an application component (the program or piece of code that performs a specific business function) can send requests to the SAP server (for example, to query a customer record in an SAP table or to update an order document) or receive events from the server (for example, to be notified that a customer record has been updated). The adapter creates a standard interface to the applications and data on the SAP server, so that the application component does not have to understand the lower-level details (the implementation of the application or the data structures) on the SAP server.

WebSphere Adapter for SAP Software complies with the Java™ 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) version 1.5. JCA 1.5 standardizes the way application components, application servers, and enterprise information systems, such as an SAP server, interact with each other. WebSphere Adapter for SAP Software makes it possible for JCA-compliant application servers to connect to and interact with the SAP server. Application components running on the JCA-compliant server can then communicate with the SAP server in a standard way (using business objects or JavaBeans™).

The following example assumes you are setting up an adapter using Rational® Application Developer for WebSphere Software and deploying the module that includes the adapter to WebSphere Application Server.

Suppose a company uses SAP Software to coordinate most of its business operations. SAP includes a business function that returns a list of customers in response to a range of customer IDs. An application component might be able to use this function as part of an overall business process. For example, the promotions department within the company sends advertising material to customers, and, as part of that process, needs to first obtain a list of customers.

The SAP function does not have a Web service interface, however, so the application component used by the promotions department would need to understand the low-level API and data structures of the SAP function in order to make the call to the function. Information technology resources and time would be needed to create the linkage between the application component and the SAP function.

With the WebSphere Adapter for SAP Software, you can automatically generate an interface to the SAP function to hide the lower-level details of the function. Depending on how you want to use the adapter, you can embed it with the deployed module, or install the adapter as a stand-alone component, to be used by more than one application. The adapter is deployed to WebSphere Application Server. The application component interacts with the adapter instead of with the SAP function.

The adapter, which you generate with the J2C Bean wizard of Rational Application Developer for WebSphere Software, uses a standard interface and standard data objects. The adapter takes the standard data object sent by the application

component and calls the SAP function. The adapter then returns a standard data object to the application component. The application component does not have to deal directly with the SAP function; it is the SAP adapter that calls the function and returns the results.

For example, the application component that needed the list of customers would send a standard business object with the range of customer IDs to the SAP adapter. The application component would receive, in return, the results (the list of customers) in the form of a standard business object. The application component would have no need to know how the function worked or how the data was structured. The adapter would perform all the interactions with the actual SAP function.

Similarly, the client application might want to know about a change to the data on the SAP server (for example, a change to a particular customer). You can generate an adapter component that listens for such events on the SAP server and notifies client applications with the update. In this case, the interaction begins at the SAP server.

Hardware and software requirements

The hardware and software requirements for WebSphere Adapters are provided on the IBM® Support Web site.

To view hardware and software requirements for WebSphere Adapters, see http://www.ibm.com/support/docview.wss?uid=swg27006249

Additional information

The following links provide additional information you might need to configure and deploy your adapter:

- The compatibility matrix for WebSphere Business Integration Adapters and WebSphere Adapters identifies the supported versions of required software for your adapter. To view this document, go to the WebSphere Adapters support page and click **Compatibility Matrix** beneath the **Related** heading in the **Additional support links** section: http://www.ibm.com/software/integration/wbiadapters/support/.
- Technotes for WebSphere Adapters provide workarounds and additional information that are not included in the product documentation. To view the technotes for your adapter, go to the following Web page, select the name of your adapter from the **Product category** list, and click the search icon: http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm.

Technical overview of WebSphere Adapter for SAP Software

WebSphere Adapter for SAP Software provides multiple ways to interact with applications and data on SAP servers. Outbound processing (from an application to the adapter to the SAP server) and inbound processing (from the SAP server to the adapter to an application) are supported.

For outbound processing, the adapter client invokes the adapter operation to create, update, or delete data on the SAP server or to retrieve data from the SAP server.

For inbound processing, an event that occurs on the SAP server is sent from the SAP server to the adapter. The ALE inbound and BAPI inbound interfaces start event listeners that detect the events. Conversely, the Advanced event processing interface polls the SAP server for events. The adapter then delivers the event to an endpoint, which is an application or other consumer of the event from the SAP server.

You configure the adapter to perform outbound and inbound processing by using the J2C Bean wizard to create a deployable module that includes the interface to the SAP application as well as business objects based on the functions or tables it discovers on the SAP server.

Overview of the outbound processing interfaces

WebSphere Adapter for SAP Software provides multiple interfaces to the SAP server for outbound processing.

- Through its BAPI interfaces, the adapter issues remote function calls (RFCs) to RFC-enabled functions, such as a Business Application Programming Interface (BAPI) function. These remote function calls create, update, or retrieve data on an SAP server .
  - The BAPI interface works with individual BAPIs (simple BAPIs). For example, you might want to check to see whether specific customer information exists in an SAP database.
  - The BAPI work unit interface works with ordered sets of BAPIs. For example, you might want to update an employee record. To do so, you use three BAPIs to lock the record (to prevent any other changes to the record), update the record, and have the record approved.
  - The BAPI result set interface uses two BAPIs to select multiple rows of data from an SAP database.

  BAPI calls are useful when you need to perform data retrieval or manipulation and a BAPI or RFC function that performs the task already exists.

  Simple BAPIs can be sent through the Synchronous RFC, Asynchronous Transactional RFC, or Asynchronous Queued RFC protocol.
  - With Synchronous RFC, both the adapter and the SAP server must be available when the call is made from the adapter to the SAP server. The adapter sends a request to the SAP server and waits for a response.
  - With Asynchronous Transactional RFC, a transaction ID is associated with the call from the adapter to the SAP server. The adapter does not wait for a response from the SAP server. Only the transaction ID is returned to the client application.
  - With Asynchronous Queued RFC, the call from the adapter is delivered to a predefined queue on the SAP server. As with Asynchronous Transactional RFC, a transaction ID is associated with the call, and the adapter does not wait for a response from the SAP server.

    This interface is useful when the event sequence must be preserved.
- The Query interface for SAP Software retrieves data from specific SAP application tables. It can return the data or check for the existence of the data. You can use this type of interaction with SAP if you need to retrieve data from an SAP table without using an RFC function or a BAPI.
- With the Application Link Enabling (ALE) interface, you exchange data using SAP Intermediate Data structures (IDocs). For outbound processing, you send an IDoc or a packet of IDocs to the SAP server.

The ALE interface, which is particularly useful for batch processing of IDocs, provides asynchronous exchange. You can use the Queued Transactional (qRFC) protocol to send the IDocs to a queue on the SAP server. The qRFC protocol ensures the order in which the IDocs are received. It is often used for system replications or system-to-system transfers.

- With the ALE pass-through IDoc interface, the adapter sends the IDoc to the SAP server with no conversion of the IDoc. The business object contains stream data representing the IDoc.
- With the Advanced event processing interface, you send data to the SAP server. The data is then processed by an ABAP handler on the SAP server.

Overview of the inbound processing interfaces

WebSphere Adapter for SAP Software provides the following interfaces to the SAP server for inbound processing.

- Through its BAPI inbound interface, the adapter listens for events and receives notifications of RFC-enabled function calls from the SAP server.
  - With Synchronous RFC, both the adapter and the SAP server must be available when the call is made from the SAP server to the adapter. The adapter sends the request to a predefined application and returns the response to the SAP server.

    Note: In version 6.1.0 of the WebSphere Adapter for SAP Software, inbound synchronous processing of RFC-enabled functions was known as the *Synchronous callback interface*.
  - With Asynchronous Transactional RFC, the event will be delivered to the adapter even if the adapter is not available when the call is made. The SAP server stores the event on a list of functions to be invoked and continues to attempt to deliver it until the adapter is available.

    Note: You also use Asynchronous Transactional RFC if you want to deliver the functions from a predefined queue on the SAP server. Delivering the files from a queue ensures the order in which the functions are sent.

    If you select assured-once delivery, the adapter uses a data source to persist the event data received from the SAP server. Event recovery is provided to track and recover events in case a problem occurs when the adapter attempts to deliver the event to the endpoint.
- With the ALE inbound processing interface, the adapter listens for events and receives one or more IDocs from the SAP server. As with ALE outbound processing, ALE inbound processing provides asynchronous exchange.

  You can use the qRFC interface to receive the IDocs from a queue on the SAP server, which ensures the order in which the IDocs are received.

  If you select assured-once delivery, the adapter uses a data source to persist the event data, and event recovery is provided to track and recover events in case a problem occurs when the adapter attempts to deliver the event to the endpoint.
- With the ALE pass-through IDoc interface, the SAP server sends the IDoc through the adapter to the endpoint with no conversion of the IDoc. The business object contains stream data representing the IDoc.
- The Advanced event processing interface polls the SAP server for events. It discovers events waiting to be processed. It then processes the events and sends them to the endpoint.

How the adapter interacts with the SAP server

The adapter uses the SAP Java Connector (SAP JCo) API to communicate with SAP applications. An application sends a request to the adapter, which uses the SAP JCo API to convert the request into a BAPI function call. The SAP system processes the request and sends the results to the adapter. The adapter sends the results in a response message to the calling application.

How the adapter is packaged

WebSphere Adapter for SAP Software is packaged and delivered as two RAR files, and the one you use depends on whether the invoked SAP function supports transactional behavior.

- If the targeted function (for example, BAPI) supports transactions, use the CWYAP_SAPAdapter_Tx.rar adapter because it supports local transaction behavior and, as such, can participate in the transaction managed by the WebSphere Application Server Transaction Manager.
- If the targeted function (for example, BAPI) does not support transactions, use the CWYAP_SAPAdapter.rar adapter because it indicates to the WebSphere Application Server Transaction Manager that the interaction performed with the SAP system cannot participate in and follow transactional semantics.

Outbound Processing

The adapter uses the SAP Java™ Connector (SAP JCo) API to communicate with SAP applications. A client application sends a request to the adapter which uses the SAP JCo API to convert the request into a function call. The SAP system processes the request and sends the results to the adapter. The adapter sends the results in a response message to the calling application.

The Adapter for SAP Software provides multiple interfaces to the SAP server for outbound processing. A summary of these interfaces is provided below:

- Through its BAPI interfaces, the adapter issues remote function calls (RFCs) to RFC-enabled functions, such as a Business Application Programming Interface (BAPI) function. These remote function calls create, update, or retrieve data on an SAP server and return the results to the calling application.
  - The BAPI interface works with individual BAPIs. For example, you might want to check to see whether specific customer information exists in an SAP database.
  - The BAPI work unit interface works with ordered sets of BAPIs. For example, you might want to update an employee record. To do so, you use three BAPIs to lock the record (to prevent any other changes to the record), update the record, and have the record approved.
  - The BAPI Result Set interface uses two BAPIs to select multiple rows of data from an SAP database.

  BAPI calls are useful when you need to perform data retrieval or manipulation and a BAPI or RFC function that performs the task already exists.
- With the Application Link Enabling (ALE) interface, you exchange data using SAP Intermediate Data structures (IDocs). For outbound processing, you send an IDoc or a packet of IDocs to the SAP server.

  The ALE interface, which is particularly useful for batch processing of IDocs, provides asynchronous exchange. You can use the queued transactional (qRFC) protocol to send the IDocs to a queue on the SAP server. The qRFC protocol

ensures the order in which the IDocs are received. It is often used for system replications or system-to-system transfers.

- The Query interface for SAP Software retrieves data from specific SAP application tables. It can return the data or check for the existence of the data. You can use this type of interaction with SAP if you need to retrieve data from an SAP table without using an RFC function or a BAPI.
- With the Advanced event processing interface, you send data to the SAP server. The data is then processed by an ABAP handler on the SAP server.

Inbound Processing

Adapter for SAP Software provides three interfaces to the SAP server for inbound processing.

- Through its BAPI inbound interface, the adapter listens for events and receives notifications of RFC-enabled function calls from the SAP server. The adapter sends the request to a predefined application and returns the response to the SAP server.
- With the ALE inbound processing interface, the adapter listens for events and receives one or more IDocs from the SAP server. As with ALE outbound processing, ALE inbound processing provides asynchronous exchange.

  You can use the qRFC interface to receive the IDocs from a queue on the SAP server, which ensures the order in which the IDocs are received.

  The adapter uses a data source to persist the event data, and event recovery is provided to track and recover events in case of abrupt termination.
- The Advanced event processing interface polls the SAP server for events. It discovers events waiting to be processed. It then processes the events and sends them to the endpoint.

Adapter Packaging

Adapter for SAP Software on WebSphere Application Server is packaged and delivered as two RAR files, and the one you use depends on whether the invoked SAP function supports transactional behavior.

- If the targeted function (for example, BAPI) supports transactions, use the CWYAP_SAPAdapter_Tx.rar adapter because it supports local transaction behavior and, as such, can participate in the transaction managed by the WebSphere Application Server Transaction Manager.
- If the targeted function (for example, BAPI) does not support transactions, use the CWYAP_SAPAdapter.rar adapter because it indicates to the WebSphere Application Server Transaction Manager that the interaction performed with the SAP system cannot participate in and follow transactional semantics.

Business objects

A business object is a structure that consists of data, the action to be performed on the data, and additional instructions, if any, for processing the data. The data can represent either a business entity, such as an invoice or an employee record, or unstructured text.

For outbound processing, the adapter client uses business objects to send data to SAP or to obtain data (through the adapter) from SAP. In other words, the client sends a business object to the adapter and the adapter converts the data in the business object to a format that is compatible with an SAP API call. The adapter then invokes the SAP API with this data.

For inbound processing, the SAP server sends a function call through the adapter to an endpoint. The adapter converts the function call into a business object.

The adapter uses the metadata that is generated by the J2C Bean wizard to construct a business-object definition. This metadata contains information such as the operation of the business object and import or export parameters.

How data is represented in business objects

The way data is represented in a business object depends on the interface to SAP that you are using.

For example, a BAPI business-object definition, which is generated by the J2C Bean wizard, is modeled on the BAPI function interface in SAP. The business-object definition represents a BAPI function.

For the ALE interface, the business-object definition is based on standard or extension IDocs available on the SAP server.

For the Query Interface for SAP Software, the data in the business object represents the columns of the associated table in SAP.

For the advanced event processing interface, business objects are based on custom IDocs, standard IDocs, or extension IDocs available on the SAP server.

How business-object definitions are created

You create business-object definitions by using the J2C Bean wizard, launched from Rational Application Developer for WebSphere Software. The wizard connects to the application, discovers data structures in the application, and generates business-object definitions to represent them. It also generates other artifacts needed by the adapter, such as the interface information that indicates the input and output parameters.

Java data bindings

The business data exchanged between the client application and the resource adapter is represented as Java Data Bindings. The metadata describing the business data is defined as business objects and represented as the XSD schemas. The Java Data Bindings are generated from these XSDs and are the realization of the business objects.

A java data binding is a structure that consists of data and, in some cases, metadata with additional instructions, for processing the data. It is a generated, hierarchical, java objects implementing Record interface. The data can represent a business entity, such as an invoice or an employee record.

You create java data bindings by using the J2C Bean wizard, launched from Rational Application Developer Connector Tools. The wizard connects to the SAP system, discovers data structures in the EIS, and generates data bindings to represent them. The adapter supports records that are hierarchically structured. Information about the processed object is stored in the application-specific information for the object and each of its attributes.

The J2C Bean wizard

The J2C Bean wizard is a tool you use to create services. The J2C Bean wizard establishes a connection to the SAP server, discovers services (based on search criteria you provide), and generates business objects, interfaces, and import or export files, based on the services discovered.

Using Rational Application Developer for WebSphere Software, you establish a connection to the SAP server to browse the metadata repository on the SAP server. The SAP metadata repository, which is a database of the SAP data, provides a consistent and reliable means of access to that data.

You specify connection information (such as the user name and password needed to access the server), and you specify the interface you want to use (for example, BAPI).

The service metadata associated with that interface is displayed. You can then provide search criteria and select the information (for example, you can list all BAPIs that begin with ″CUSTOMER″ and then select one or more BAPIs).

The result of running the J2C Bean wizard is a module that contains the interfaces and business objects along with the adapter. You deploy this module on WebSphere Application Server

For example, if you run the J2C Bean wizard and select BAPI_CUSTOMER_GETLIST, you see, under **Data Types**, a list of generated business objects, including the objects associated with any faults that might be generated during processing.

The interface, showing the input and output parameters for the operation as well as the types of business objects used, is also generated.

The J2C Bean wizard also produces an import file (for outbound processing) or an export file (for inbound processing).

- The import file contains the managed connection factory property settings you provided in the wizard.
- The export file contains the activation specification property settings you provided in the wizard.

Standards Compliance

This product is compliant with several government and industry standards, including accessibility standards and Internet protocol standards.

Accessibility

Administration

The run time administrative console is the primary interface for deployment and administration of enterprise applications. These consoles are displayed within a standard web browser. By using an accessible Web browser, such as Microsoft® Internet Explorer or Netscape Browser, you are able to:

- Use screen-reader software and a digital speech synthesizer to hear what is displayed on the screen
- Use voice recognition software, such as IBM via Voice, to enter data and to navigate the user interface

- Operate features by using the keyboard instead of the mouse

You can configure and use product features by utilizing standard text editors and scripted or command line interfaces instead of the graphical interfaces that are provided. When appropriate, the documentation for specific product features contains additional information about the accessibility of the features.

J2C Bean wizard
The J2C Bean wizard is the primary component used to create application accessing EIS systems. This wizard is implemented as an Eclipse plug-in that is available through Rational Application Developer for WebSphere Software is fully accessible.

Keyboard navigation
This product uses standard Microsoft Windows® navigation keys.

IBM and accessibility
See the IBM Accessibility Center web site http://www.ibm.com/able/ for more information about the commitment that IBM has to accessibility.

Internet Protocol, Version 6 (IPv6)

IBM WebSphere Application Server, version 6.1.0 and later and its JavaMail component support dual-stack Internet Protocol Version 6.0 (IPv6). For more information about this compatibility in WebSphere Application Server, see IPv6 support in the WebSphere Application Server information center. For more information about IPv6, see http://www.ipv6.org.

Chapter 2. Planning for adapter implementation

Before you configure WebSphere Adapter for SAP Software, consider whether you will set up the adapters in a clustered environment, in which the workload of the server is distributed across multiple machines.

Before you begin

Before you begin to set up and use the adapter, you should possess a thorough understanding of business integration concepts, the capabilities and requirements of the integration development tools and runtime environment you will use, and theSAP server environment where you will build and use the solution.

To configure and use WebSphere Adapter for SAP Software, you should understand and have experience with the following concepts, tools, and tasks:

- The business requirements of the solution you are building.
- The capabilities provided by the integration development tools you will use to build the solution. You should know how to use these tools to create modules, test components, and complete other integration tasks.
- The capabilities and requirements of the runtime environment you will use for the integration solution. You should know how to configure and administer the host server and how to use the administrative console to set and modify property definitions, configure connections, and manage events.
- The SAP server environment in which you are working. This includes a detailed understanding of the SAP GUI, RFC-enabled functions (such as BAPIs), and ALE IDocs.

Security

The adapter uses the J2C authentication data entry, or authentication alias, feature of Java 2 security to provide secure user name and password authentication. For more information about security features, see the documentation for WebSphere Application Server. The adapter also supports secure network connections for both outbound and inbound processing.

Support for protecting sensitive user data in log and trace files

The adapter provides the ability to prevent sensitive or confidential data in log and trace files from being seen by those without authorization.

Log and trace files for the adapter can contain data from your SAP server, which might contain sensitive or confidential information. Sometimes these files must be seen by individuals without authorization to view sensitive data. For example, a support specialist must use the log and trace files to troubleshoot a problem.

To protect the data in situations like this, the adapter lets you specify whether or not you want to prevent confidential user data from displaying in the adapter log and trace files. You can select this option in the J2C Bean wizard or change the HideConfidentialTrace property. When this property is enabled, the adapter replaces the sensitive data with XXX's.

See “Managed connection factory properties” on page 176 for information about this optional property.

The following types of information are considered potentially sensitive data and are disguised:

- The contents of a business object
- The contents of the object key of the event record
- User name, Password, Environment, and Role
- The URL used to connect to the SAP server
- Business object data in an intermediate form, such as the fields in a BAPI

The following types of information are not considered user data and are not hidden:

- The contents of the event record that are not part of the event record object key, for example, the XID, event ID, business object name, and event status
- Business object schemas
- Transaction IDs
- Call sequences

User authentication

The adapter supports several methods for supplying the user name and password that are needed to connect to the SAP server. By understanding the features and limitations of each method, you can pick a method that provides the appropriate level of security and convenience for your application.

To integrate an adapter into your application, a user name and password are needed at the following times:

- When the J2C Bean wizard connects to the SAP server to extract, or *discover*, information about the objects and services that you can access with the adapter.
- At run time on WebSphere Application Server, when the adapter connects to the SAP server to process outbound requests and inbound events.

Authentication in the wizard

The J2C Bean wizard asks for connection information for the discovery process, and then reuses it as the default values of the adapter properties that specify the connection information used at run time. You can use a different user name and password while running the wizard than you use when the application is deployed to the server. You can even connect to a different SAP server, although the schema name must be the same in both databases. For example, while developing and integrating an application that uses Adapter for SAP Software, you might not use the production database; using a test database with the same data format but fewer, simulated records lets you develop and integrate the application without impacting the performance of a production database and without encountering restrictions caused by the privacy requirements for customer data.

The wizard uses the user name and password that you specify for the discovery process only during the discovery process; they are not accessible after the wizard completes.

Authentication at run time

At run time, the adapter needs to provide the user name and password to connect to the SAP server. To connect without user intervention, the adapter must access a saved copy of the user information. In a server environment, there are several methods for saving user information. The J2C Bean wizard lets you configure the adapter to get the user information using any of the following methods:

- Adapter properties
- J2C authentication alias

Saving the user name and password in adapter properties is a direct way to provide this information at run time. You provide this user name and password when you use the J2C Bean wizard to configure your module. Although directly specifying the user name and password seems the most straightforward method, it has important limitations. Adapter properties are not encrypted; the password is stored as clear text in fields that are accessible to others on the server. Also, when the password changes, you must update the password in all instances of the adapter that access that SAP server. This includes the adapters embedded in application EAR files as well as adapters that are separately installed on the server.

Using a J2C authentication data entry, or authentication alias, created with the Java Authentication and Authorization Service (JAAS) feature of Java 2 security is a robust, secure way to deploy applications. An administrator creates the authentication alias that is used by one or more applications that need to access a system. The user name and password can be known only to that administrator, who can change the password in a single place when a change is required.

Deployment options

There are two ways to deploy the adapter. You can either embed it as part of the deployed application, or you can deploy it as a stand-alone RAR file. The requirements of your environment will affect the type of deployment option you choose.

The deployment options are described below:

- When you deploy the adapter as an embedded component, the adapter is included within an enterprise application archive (EAR) file and is available only to the application in the EAR file.
- When you deploy the adapter as a stand-alone component, the adapter is represented by a stand-alone resource adapter archive (RAR) file. When it is deployed, it is available to all applications deployed in the server instance.

While creating the project for your application using Rational Application Developer for WebSphere Software, you can choose how to package the adapter [either bundled with the (EAR) file or as a stand-alone (RAR) file]. Your choice will affect how the adapter is used in the runtime environment, as well as how the properties for the adapter are displayed on the administrative console.

Choosing either to embed an adapter with your application or to deploy the adapter as a stand-alone module depends on how you want to administer the adapter. If you want a single copy of the adapter and do not care about disruption to multiple applications when you upgrade the adapter, then you would be more likely to deploy the adapter as a stand-alone module.

If you plan on running multiple versions, and if you care more about potential disruption when you upgrade the adapter, you would be more likely to embed the adapter with the application. Embedding the adapter with the application allows

you to associate an adapter version with an application version and administer it as a single module. To deploy the RAR file to the application server, you must obtain and install Adapter for SAP Software (product number 5724-T72). This provides the RAR file that you install following instructions supplied with WebSphere Application Server.

Considerations for embedding an adapter in the application

Take into consideration the following items if you plan on embedding the adapter with your application:

- An embedded adapter has class loader isolation.

  A class loader affects the packaging of applications and the behavior of packaged applications deployed on runtime environments. *Class loader isolation* means the adapter cannot load classes from another application or module. Class loader isolation prevents two similarly named classes in different applications from interfering with each other.
- Each application in which the adapter is embedded must be administered separately.

Considerations for using a stand-alone adapter

Take into consideration the following items if you plan on using a stand-alone adapter:

- Stand-alone adapters have no class loader isolation.

  Because stand-alone adapters have no class loader isolation, only one version of any given Java artifact is run and the version and sequence of that artifact is undetermined. For example, when you use a stand-alone adapter there is only *one* resource adapter version, *one* adapter foundation class (AFC) version, or *one* third-party JAR version. All adapters deployed as stand-alone adapters share a single AFC version, and all instances of a given adapter share the same code version. All adapter instances using a given third-party library must share that library.
- If you update any of these shared artifacts, all applications using the artifacts are affected.

  For instance, if you have an adapter that is working with server version X, and you update the version of the client application to version Y, your original application might stop working.
- Adapter Foundation Classes (AFC) is compatible with previous versions, but the latest AFC version must be in every RAR file that is deployed in a stand-alone manner.

  If more than one copy of any JAR file is in the class path in a stand-alone adapter, the one that is used is random; therefore, they all must be the latest version.

WebSphere Adapters in clustered environments

You can improve adapter performance and availability by deploying the module to a clustered server environment. The module is replicated across all servers in a cluster, regardless of whether you deploy the module using a stand-alone or embedded adapter.

WebSphere Application Server, WebSphere Application Server Network Deployment, and WebSphere Extended Deployment support clustered environments. Clusters are groups of servers that are managed together to balance

workloads and to provide high availability and scalability. When you set up a server cluster, you create a Deployment Manager profile. The HAManager, a subcomponent of the Deployment Manager, notifies the Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) container to activate the adapter instance. The JCA container provides a runtime environment for adapter instances.

Using WebSphere Extended Deployment, you can optionally enhance the performance of adapter instances in your clustered environment. WebSphere Extended Deployment extends the WebSphere Application Server Network Deployment capabilities by using a dynamic workload manager instead of a static workload manager, which is used by WebSphere Application Server Network Deployment. The dynamic workload manager can optimize the performance of adapter instances in the cluster by dynamically balancing the load of the requests. This means that application server instances can be automatically stopped and started based on the load variations, allowing machines with different capacities and configurations to evenly handle load variations.

In clustered environments, adapter instances can handle both inbound and outbound processes.

High availability for inbound processes

Inbound processes are based on events triggered as a result of updates to data in the SAP server. WebSphere Adapter for SAP Software is configured to detect updates through event listeners or by polling an event table. The adapter then publishes the event to its endpoint.

When you deploy a module to a cluster, the Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) container checks the enableHASupport resource adapter property. If the value for the enableHASupport property is true, which is the default setting, all of the adapter instances are registered with the HAManager with a policy 1 of N. This policy means that only one of the adapter instances starts polling or listening for events. Although other adapter instances in the cluster are started, they remain dormant with respect to the active event until the active adapter instance finishes processing the event. If the server on which the polling thread was started shuts down for some reason, an adapter instance that is running on one of the backup servers is activated.

Important: Do not change the setting of the enableHASupport property.

High availability for outbound processes

In clustered environments, multiple adapter instances are available to perform outbound process requests. Accordingly, if your environment has multiple applications that interact with WebSphere Adapter for SAP Software for outbound requests, then you might improve performance by deploying the module to a clustered environment. In a clustered environment, multiple outbound requests can be processed simultaneously, as long as they are not attempting to process the same record.

If multiple outbound requests are attempting to process the same record, such as a Customer address, the workload management capability in WebSphere Application Server Network Deployment distributes the requests among the available adapter instances in the sequence they were received. As a result, these types of outbound requests in a clustered environment are processed in the same manner as those in a

single server environment: one adapter instance processes only one outbound request at a time.

Chapter 3. SAP interfaces

The SAP interfaces enable business process integration and asynchronous data communication between two or more SAP systems or between SAP and external systems.

The following interfaces exist for SAP software:

- BAPI interfaces
- BAPI work unit interface
- BAPI result set interface
- ALE interface
- ALE pass-through IDOC interface
- Query interface
- Advanced event processing interface

The BAPI interfaces

The WebSphere Adapter for SAP Software supports outbound processing and inbound processing for simple BAPIs. In outbound processing, client applications call BAPIs and other RFC-enabled functions on the SAP server. In inbound processing, the SAP server sends an RFC-enabled function (such as a BAPI function) through the adapter to an endpoint.

Suppose you want to build a service that creates a new customer on the SAP server. You first run the external service wizard to discover the BAPI_CUSTOMER_CREATEFROMDATA function. The wizard then generates the business object definition for BAPI_CUSTOMER_CREATEFROMDATA along with other artifacts. During BAPI outbound processing, the adapter receives the service request and converts the data into a BAPI invocation.

A simple BAPI performs a single operation, such as retrieving a list of customers. The adapter supports simple BAPI calls by representing each with a single business object schema.

Simple BAPIs can be used for outbound or inbound processing. You can specify synchronous RFC processing or asynchronous transactional RFC (tRFC) processing when you configure a module for a simple BAPI. In addition, for outbound processing, you can specify asynchronous queued RFC (qRFC) processing, in which BAPIs are delivered to a predefined queue on the SAP server.

- In synchronous RFC processing, the SAP server and the adapter must be available during processing.
  - In outbound processing, the client application sends a request and then waits for a response from the SAP server.
  - In inbound processing, the SAP server sends a request through the adapter to an endpoint and waits for a response from the adapter.
- In asynchronous tRFC outbound processing, the adapter associates a transaction ID with the function call to the SAP server. The adapter does not wait for a response from the SAP server. If the delivery is unsuccessful, the client application can use the TID to make the request again.

- In asynchronous tRFC inbound processing, the adapter does not have to be available when the SAP server invokes the function call. The function call is placed on a list of functions to be invoked, and the call is attempted until it is successful.

  To send function calls from a user-defined outbound queue on the SAP server, you also specify asynchronous tRFC inbound processing.
- In asynchronous qRFC outbound processing, the process is similar to asynchronous tRFC outbound processing. A TID is associated with the function call, and the adapter does not wait for a response from the SAP server. Additionally, the BAPIs are delivered to a predefined queue on the SAP server. By sending BAPIs to the predefined queue, you can ensure the order in which they are delivered.

Outbound processing for the BAPI interface

In BAPI outbound processing, a client application sends a request to the SAP server. For simple BAPIs, you can request that processing be handled synchronously or asynchronously (the client application does not wait for a response from the SAP server).

You make a selection, during configuration, about the type of remote RFC call you want to make.

Synchronous RFC

If you select **Synchronous RFC** (the default) during configuration for a simple BAPI, the following processing steps occur:

1. The adapter receives a request from a client application in the form of a BAPI business object.
2. The adapter converts the BAPI business object to an SAP JCo function call.
3. The adapter uses the Remote Function Call (RFC) interface to process the BAPI or RFC function call in the SAP application.
4. After passing the data to the SAP server, the adapter handles the response from SAP and converts it back into the business object format required by the client application.
5. The adapter then sends the response back to the client application.

Asynchronous transactional RFC

If you select **Asynchronous transactional RFC** during configuration, the following processing steps occur:

1. The adapter receives a request from a client application in the form of a BAPI business object.
2. The adapter checks the business object to see whether the SAPTransactionID attribute has a value assigned.
   - If the SAPTransactionID has a value, the adapter uses that value during processing.
   - If the attribute does not have a value, the adapter makes a call to the SAP server and gets a transaction ID from the SAP server.
3. The adapter converts the BAPI business object to an SAP JCo function call.
4. The adapter uses the transactional Remote Function Call (tRFC) protocol to make the call to the SAP server .

   The adapter does not wait for a response from the SAP server.

5. After the function data is passed to the SAP application, control returns to the adapter.
   - If the call to the SAP server fails, the SAP server throws an ABAPException.
   - If the call to the SAP server succeeds but contains invalid data, no exception is returned to the adapter. For example, if the adapter sends a request that contains an invalid customer number, the adapter does not respond with an exception indicating that no such customer exists.
6. The adapter passes the TID information to the client.

Asynchronous queued RFC

If you select **Asynchronous queued RFC** during configuration, the following processing steps occur:

1. The adapter receives a request from a client application in the form of a BAPI business object.
2. The adapter checks the business object to see whether the SAPTransactionID attribute has a value assigned.
   - If the SAPTransactionID has a value, the adapter uses that value during processing.
   - If the attribute does not have a value, the adapter makes a call to the SAP server and gets a transaction ID from the SAP server.
3. The adapter converts the BAPI business object to an SAP JCo function call.
4. The adapter uses the tRFC protocol to make the call to the specified queue on the SAP server .

   The adapter does not wait for a response from the SAP server.
5. After the function data is passed to the SAP application, control returns to the adapter.
   - If the call to the SAP server fails, the SAP server throws an ABAPException.
   - If the call to the SAP server succeeds but contains invalid data, no exception is returned to the adapter. For example, if the adapter sends a request that contains an invalid customer number, the adapter does not respond with an exception indicating that no such customer exists.
6. The adapter passes the TID information to the client.

Inbound processing for the BAPI interface

The adapter supports inbound processing (from the SAP server to the adapter) for simple BAPIs. A client application on the SAP server invokes a function through the adapter to an endpoint.

Synchronous and asynchronous RFC

For BAPI inbound processing, you can specify that the processing be handled synchronously (in which both the client application and the adapter must be available during processing) or asynchronously (in which the adapter does not have to be available when the client application invokes the function call). In synchronous processing, the client application waits for a response from the adapter. In asynchronous processing, the client application does not wait for a response.

The BAPI interface has two sets of activation specification properties (one for synchronous RFC and one for asynchronous RFC), which you use to set up inbound processing. You specify values for the properties with the J2C Bean wizard or through the administrative console.

The sequence of processing actions that result from an inbound request differ, depending on the selection you make during configuration from the **SAP Remote Function Call (RFC) type** list.

Synchronous RFC

If you select **Synchronous RFC** (the default) during configuration, the following processing steps occur:

1. The adapter starts event listeners, which listen for an RFC-enabled function event (which you specified with the RFCProgramID property) on the SAP server.
2. The RFC-enabled function event is pushed to the adapter by way of the event listeners.
3. The adapter resolves the operation and business object name using the received RFC-enabled function name.
4. The adapter sends the business object to an endpoint in a synchronous manner.
5. The adapter receives the response business object from the endpoint.
6. The adapter maps the response business object to an RFC-enabled function and returns it to the SAP server.

The adapter does not listen for events until the endpoint is active and available.

Asynchronous transactional RFC

If you select **Asynchronous Transactional/Queued RFC** during configuration, the following processing steps occur:

1. A client on the SAP server invokes an RFC-enabled function call on the adapter.

   Note: To send the RFC-enabled functions from a queue on the SAP server, the client program on the SAP server delivers the events to a user-defined outbound queue.

   A transaction ID is associated with the call.

   The calling program on the SAP server does not wait to see whether the call to the adapter was successful, and no data is returned to the calling program.
2. The RFC-function call is placed on a list of functions to be delivered.

   You can see the event list by entering transaction code SM58 on the SAP server
3. The RFC-function call is invoked on the adapter. If the adapter is not available, the call remains in the list on the SAP server, and the call is repeated at regular intervals until it can be processed by the adapter.

   When the SAP server successfully delivers the call event, it removes the function from the list.
4. If you selected **Ensure once-only event delivery**, the adapter sets the transaction ID in the event persistent table.

   This is to ensure the event is not processed more than once.
5. The adapter resolves the operation and business object name using the received RFC-enabled function name.
6. The adapter sends the business object to an endpoint.

   If you are sending functions from a user-defined queue on the SAP server, the functions are delivered in the order in which they exist on the queue. You can see the contents of the queue by entering transaction code SMQ1 on the SAP server.

7. If the delivery is successful, and if you selected **Ensure once-only event delivery**, the adapter removes the transaction ID from the event persistent table.

   If there is a failure when the adapter attempts to deliver the business object, the transaction ID remains in the event table. When another event is received from the SAP server, the following processing occurs:
   a. The adapter checks the transaction ID.
   b. If the event matches an ID in the table, the adapter processes the failed event once.

      In other words, it does not send the event with the duplicate ID, thereby ensuring that the event is processed only once.

Event recovery

You can configure the adapter for BAPI inbound processing so that it supports event recovery in case a failure occurs while the event is being delivered from the adapter to the endpoint. When event recovery is specified, the adapter persists the event state in an event recovery table that resides on a data source.

Event recovery is not the default; you must specify it by enabling once-only delivery of events during adapter configuration. You must also set up the data source before you can create the event recovery table.

Data source

Event recovery for BAPI inbound processing requires that a JDBC data source be configured. You use the administrative console to configure the data source. You select a JDBC provider (for example, Derby) and then create a new data source.

Event recovery table

You can create the event recovery table manually, or you can have the adapter create the event table. The value of the EP_CreateTable configuration property determines whether the event recovery table is created automatically. The default value of this property is `True` (create the table automatically).

To create the table manually, use the information provided in the following table.

Table 1. Event recovery table fields

Table field name	Type	Description
EVNTID	VARCHAR(255)	Transaction ID for the tRFC (Transactional Remote Function Call) protocol. The tRFC protocol significantly improves the reliability of the data transfer, but it does not ensure that the order of BAPI transactions specified in the application is observed. Event ordering is also affected by the number of event listeners. However, at some point all BAPI transactions are transferred.
EVNTSTAT	INTEGER	Event processing status. Possible values are: • 0 (Created) • 1 (Executed) • 3 (In Progress) • -1 (Rollback)

Table 1. Event recovery table fields (continued)

Table field name	Type	Description
XID	VARCHAR(255)	An XA resource keeps track of transaction IDs (XIDs) in the event recovery table. The adapter queries and updates that XID field. During recovery, WebSphere Application Server calls the resource adapter, querying it for XA resources, and then does transaction recovery on them. **Note:** The XA resource is used to enable assured once delivery. Make sure the activation specification property Assured Once Delivery is set to `true`.
BQTOTAL	INTEGER	Not used for BAPI inbound processing.
BQPROC	INTEGER	Not used for BAPI inbound processing.
EVNTDATA	VARCHAR(255)	Not used.

Java data bindings for the BAPI interface

A BAPI Java data binding, which is generated by the J2C Bean wizard, is modeled on the BAPI function interface in SAP.

A Java data binding is a structure that consists of data, the action to be performed on the data and additional instructions for processing the data. The client application uses Java data bindings to send data to or obtain data from the SAP server.

The adapter uses the BAPI metadata that is generated by the J2C Bean wizard to construct the Java data bindings. This metadata contains BAPI-related information such as the operation of the Java data binding, table parameters, transaction information, and dependent or grouped BAPIs.

You create Java data bindings by using the J2C Bean wizard, launched from Rational Application Developer for WebSphere Software. The wizard connects to the application, discovers data structures in the application, and generates Java data bindings to represent them. It also generates other artifacts needed by the adapter, such as the interface information that indicates the input and output parameters.

The structure of a BAPI Java data binding depends on the interface type (simple BAPI, BAPI work unit, or BAPI result set).

Business object structure for a simple BAPI

A business object for a simple BAPI call reflects a BAPI method or function call in SAP. Each business object property maps to a BAPI parameter. The metadata of each business-object property indicates the corresponding BAPI parameter. The operation metadata determines the correct BAPI to call.

For a simple BAPI that performs Create, Update, Retrieve, and Delete operations, each operation is represented by a business object, with the business objects being grouped together within a wrapper.

Note: The business object wrapper can be associated with multiple operations, but for a simple BAPI, each business object is associated with only one operation. For

example, while a wrapper business object can contain BAPIs for Create and Delete operations, BAPI_CUSTOMER_CREATE is associated with the Create operation, not the Delete operation.

The BAPI business objects are children of the business object wrapper, and, depending on the operation to be performed, only one child object in this wrapper needs to be populated at run time in order to process the simple BAPI call. Only one BAPI, the one that is associated with the operation to be performed, is called at a time.

If you selected **Asynchronous Transactional RFC** (for outbound or inbound processing) or **Asynchronous Queued RFC** (for outbound processing) , the BAPI wrapper business object also contains a transaction ID. The transaction ID is used to resend the BAPI call if the receiving system is not available at the time of the initial call.

Note: This object, which contains the results of the BAPI operation, is named according to the convention `Sap + Name of the structure`. If the module contains more than one SapReturn business object, a unique number is suffixed to the business object names to make them unique (for example, ″SapReturn619647890″).

Additional information about the business object can be found in the application-specific information of the business object. For example, the application-specific information for a top-level object lists the type of BAPI and operation information.

Java data binding structure for a simple BAPI

A Java data binding for a simple BAPI call reflects a BAPI method or function call in SAP.

Each Java data binding property maps to a BAPI parameter. The metadata of each Java data binding property indicates the corresponding BAPI parameter. The operation metadata determines the correct BAPI to call.

For a simple BAPI that performs Create, Update, Retrieve, and Delete operations, each operation is represented by a Java data binding, with the Java data bindings being grouped together within a wrapper Java data binding.

Note: The data binding definition can be associated with multiple operations, but only one operation is processed at run time. Each Java data binding is a child of the wrapper and represents a complex property of the wrapper.

The BAPI Java data bindings are children of the Java data binding wrapper and, depending on the operation to be performed; only one child data binding in this wrapper needs to be populated at run time in order to process the simple BAPI call. Only one BAPI, the one that is associated with the operation to be performed, is called at a time

The SapReturn Java data binding, which contains the results of the BAPI operation, is named according to the convention **Sap + Name** of the structure. If, for some reason, this structure is not named `Return` in the BAPI, the generated Java data binding will have a different name.

Additional information about the Java data binding can be found in the application-specific information of the schema which defines the Java data binding. For example, the application-specific information for a top-level object lists the type of BAPI and operation information.

Business object structure for a nested BAPI

A nested BAPI business object contains structure parameters that can contain one or more other structures as components.

Java data binding structure for a nested BAPI

A nested BAPI Java data binding contains structure parameters that can contain one or more other structures as components.

The BAPI work unit interface

The WebSphere Adapter for SAP Software supports outbound processing for BAPI units of work. A BAPI work unit consists of a set of BAPIs that are processed in sequence to complete a task.

For example, to update an employee record in the SAP system, the record needs to be locked before being updated. This is accomplished by calling three BAPIs, in sequence, in the same work unit. The following three BAPIs illustrate the kind of sequence that forms such a work unit:

- BAPI_ADDRESSEMP_REQUEST
- BAPI_ADDRESSEMP_CHANGE
- BAPI_ADDRESSEMP_APPROVE

The first BAPI locks the employee record, the second updates the record, and the third approves the update. The advantage of using a BAPI work unit is that the client application can request the employee record change with a single call, even though the work unit consists of three separate functions. In addition, if SAP requires that the BAPIs be processed in a specific sequence for the business flow to complete correctly, the work unit supports this sequence.

Outbound processing for the BAPI work unit interface

In BAPI work unit outbound processing, a client application sends a request to the SAP server. Processing is handled synchronously (the client application waits for a response from the SAP server).

For BAPI work units, the following processing steps occur:

1. The adapter receives a request from a client application in the form of a BAPI business object.
2. The adapter converts the BAPI business object to an SAP JCo function call.
3. The adapter uses the Remote Function Call (RFC) interface to process the BAPI or RFC function call in the SAP application.
4. After passing the data to the SAP server, the adapter handles the response from SAP and converts it back into the business object format required by the client application.
5. The adapter then sends the response back to the client application.

Business object structure for a BAPI work unit

A business object that represents a BAPI work unit (also known as a BAPI transaction) is actually a wrapper object that contains multiple child BAPI objects. Each child BAPI object within the wrapper object represents a simple BAPI.

The adapter supports a BAPI work unit using a top-level wrapper business object that consists of multiple child BAPIs, each one representing a simple BAPI in the sequence. The BAPI wrapper object represents the complete work unit, while the child BAPI objects contained within the BAPI wrapper object represent the individual operations that make up the work unit.

The adapter uses the sequence of operations in the operation metadata to process the BAPIs in the work unit.

Each second-level child business object represents a structure parameter or table parameter of the method. Simple attributes correspond to simple parameters of the method.

Additional information about the business object can be found in the application-specific information of the business object. For example, the application-specific information for a BAPI work unit lists the type of BAPI and the operations that make up the work unit.

Note: The adapter does not provide an automated rollback mechanism for BAPI work units. Rollback of a BAPI work unit can be achieved in one of the following ways:

- Do not put explicit COMMITs in the application-specific information sequence. When an error occurs in one of the BAPIs, the sequence of BAPI calls is terminated and BAPI_TRANSACTION_ROLLBACK is called. If there is no intrinsic COMMIT in any of the BAPIs already called, no further steps are required. Most BAPIs do not have an intrinsic COMMIT.
- Call another BAPI that can compensate for the work that is already committed, as in the case of the BAPIs that have an intrinsic COMMIT.

Java data binding structure for a BAPI work unit

A Java data binding that represents a BAPI work unit (also known as a BAPI transaction) is actually a wrapper data binding that contains multiple child BAPI data bindings. Each child BAPI data binding within the wrapper data binding represents a simple BAPI.

The adapter supports a BAPI work unit using a top-level wrapper Java data binding that consists of multiple child BAPIs, each one representing a simple BAPI in the sequence. The BAPI wrapper data binding represents the complete work unit while the child BAPI data bindings contained within the BAPI wrapper data binding represent the individual operations that make up the work unit.

The adapter uses the sequence of operations in the operation metadata to process the BAPIs in the units of work.

Note: The adapter does not provide an automated rollback mechanism for BAPI units of work. Rollback of a BAPI work unit can be achieved in one of the following ways:

- Do not put explicit COMMITs in the application-specific information sequence. When an error occurs in one of the BAPIs, the sequence of BAPI calls is

terminated and BAPI_TRANSACTION_ROLLBACK is called. If there is no intrinsic COMMIT in any of the BAPIs already called, no further steps are required. Most BAPIs do not have an intrinsic COMMIT.

- Call another BAPI that can compensate for the work that is already committed, as in the case of the BAPIs that have an intrinsic COMMIT.

The BAPI result set interface

The WebSphere Adapter for SAP Software supports outbound processing for BAPI result sets. In outbound processing, client applications call BAPIs and other RFC-enabled functions on the SAP server.

BAPI result set interface

BAPI result sets use the GetList and GetDetail functions to retrieve an array of data from the SAP server. The information returned from the GetList function is used as input to the GetDetail function.

For example, if you want to retrieve information on a set of customers, you use BAPI_CUSTOMER_GETLIST, which acts as the query BAPI, and BAPI_CUSTOMER_GETDETAIL, which acts as the result BAPI. The BAPIs perform the following steps:

1. The BAPI_CUSTOMER_GETLIST call returns a list of keys (for example, CustomerNumber).
2. Each key is mapped dynamically to the business object for BAPI_CUSTOMER_GETDETAIL.
3. BAPI_CUSTOMER_GETDETAIL is processed multiple times, so that an array of customer information is returned.

You use the J2C Bean wizard to discover the BAPI_CUSTOMER_GETLIST and BAPI_CUSTOMER_GETDETAIL functions and to build the key relationship between the two BAPIs. The wizard then generate business object definitions for these BAPIs along with other artifacts. At run time, the client sets the values in the BAPI_CUSTOMER_GETLIST business object, and the adapter returns the corresponding set of customer detail records from the SAP server.

Outbound processing for the BAPI result set interface

In BAPI result set outbound processing, a client application sends a request to the SAP server. Processing is handled synchronously (the client application waits for a response from the SAP server).

For BAPI result sets, the following processing steps occur:

1. The adapter receives a request from a client application in the form of a BAPI business object.
2. The adapter converts the BAPI business object to an SAP JCo function call.
3. The adapter uses the Remote Function Call (RFC) interface to process the BAPI or RFC function call in the SAP application.
4. After passing the data to the SAP server, the adapter handles the response from SAP and converts it back into the business object format required by the client application.
5. The adapter then sends the response back to the client application.

Business object structure for a BAPI result set

The top-level business object for a result set is a wrapper that contains a GetDetail business object. The GetDetail business object contains the results of a query for SAP data. The GetDetail business object also contains, as a child object, the query business object. The query business object represents a GetList BAPI. These two BAPIs work together to retrieve information from the SAP server.

Additional information about the business object can be found in the application-specific information of the business object. For example, the application-specific information for SapBapiCustomerGetdetail lists the type of BAPI and operation information.

Java data binding structure for a BAPI result set

The Java data binding for a result set contains a Java data binding which contains the results of a query for SAP data and the query Java data binding. These two BAPIs work together to retrieve information from the SAP server.

The top-level Java data binding for a result set is a wrapper that contains a **GetDetail** Java data binding. The **GetDetail** Java data binding contains the results of a query for SAP data. The **GetDetail** Java data binding also contains, as a child object, the query Java data binding. The query Java data binding represents a **GetList BAPI**. These two BAPIs work together to retrieve information from the SAP server.

Additional information about the Java data binding can be found in the application-specific information of the Java data binding. For example, the application-specific information for **SapBapiCustomerGetdetail** lists the type of BAPI and operation information.

The ALE interfaces

The SAP ALE interface enables business process integration and asynchronous data communication between two or more SAP systems or between SAP and external systems. The data is exchanged in the form of Intermediate Documents (IDocs).

The adapter supports outbound and inbound processing by enabling the exchange of data in the form of business objects.

- For inbound processing, SAP pushes the data in IDocs to the SAP adapter. The adapter converts the IDocs to business objects and delivers them to the endpoint.
- For outbound processing, the SAP adapter converts the business object to an IDoc and delivers it to SAP.

To use the ALE interface for inbound processing, you must make sure that your SAP server is properly configured (for example, you must set up a partner profile and register a program ID to listen for events).

Application systems are loosely coupled in an ALE integrated system, and the data is exchanged asynchronously.

IDocs

IDocs are containers for exchanging data in a predefined (structured ASCII) format across system boundaries. The IDoc type indicates the SAP format that is to be

used to transfer the data. An IDoc type can transfer several message types (the logical messages that correspond to different business processes). IDocs are used for outbound and inbound processing.

For example, if an application developer wants to be notified when a sales order is created on the SAP server, the developer runs the J2C Bean wizard to discover the ORDERS05 IDoc on the SAP server. The wizard then generates the business object definition for ORDERS05. The wizard also generates other artifacts needed to build the application.

IDocs are exchanged for inbound and outbound events, and IDocs can be exchanged either as individual documents or in packets.

For outbound processing, the adapter uses the IDoc business object to populate the appropriate RFC-enabled function call made to the SAP server.

For inbound processing, IDocs can be sent from the SAP server as parsed or unparsed documents

- For parsed documents, the adapter parses the IDoc and creates a business object that reflects the internal structure of the IDoc.
- For unparsed IDocs, the adapter processes the IDoc but does not convert the data portion of the IDoc.

Transactional RFC processing

The adapter uses tRFC (transactional RFC) to guarantee delivery and to ensure that each IDoc is exchanged only once with SAP. The tRFC component stores the called RFC function in the database of the SAP system along with a unique transaction identifier (TID).

The most common reason for using transaction ID support is to ensure once and only once delivery of data. To make sure of this feature, select the transaction RAR file (CWYAP_SAPAdapter_Tx.rar) when you configure the adapter.

Note: The SAP transaction ID property is always generated by the J2C Bean wizard; however, it is supported only for outbound operations when the CWYAP_SAPAdapter_Tx.rar version of the adapter is used.

The client application must determine how to store the SAP transaction ID and how to relate the SAP transaction ID to the data being sent to the adapter. When the events are successful, the client application should not resubmit the event associated with this TID again to prevent the processing of duplicate events.

- If the client application does not send an SAP transaction ID with the business object, the adapter returns one after executing the transaction.
- If the client application has an SAP transaction ID, it needs to populate the SAP transaction ID property with that value before executing the transaction.

The SAP transaction ID can be used for cross-referencing with a global unique ID that is created for an outbound event. The global unique ID is something you can create for managing integration scenarios.

Queued RFC processing

The adapter uses qRFC (queued transactional RFC) to ensure that IDocs are delivered in sequence to a queue on the SAP server or are received in sequence from the SAP server.

Outbound processing for the ALE interfaces

The adapter supports outbound processing (from the adapter to the SAP server) for the ALE interface. ALE uses IDocs for data exchange, and the adapter uses business objects to represent the IDocs.

The following list describes the sequence of processing actions that result from an outbound request using the ALE interface.

Note: The client application that makes the request uses the interface information that was generated by the J2C Bean wizard.

1. The adapter receives a request, which includes an IDoc business object, from a client application.
2. The adapter uses the IDoc business object to populate the appropriate RFC-enabled function call used by the ALE interface.
3. The adapter establishes an RFC connection to the ALE interface and passes the IDoc data to the SAP system. If you are using the qRFC protocol, the adapter passes the IDoc data in the order specified in the wrapper business object to the specified queue on the SAP server.
4. After passing the data to SAP, the adapter performs one of the following steps:
   - If the call is not managed by a J2C local transaction, the adapter releases the connection to SAP and does not return any data to the caller. When no exceptions are raised, the outbound transaction is considered successful. You can verify whether the data is incorporated into the SAP application by inspecting the IDocs that have been generated in SAP.
   - If the call is managed by a J2C local transaction, the adapter returns the transaction ID.

     The adapter uses the tRFC protocol to support J2C local transactions.

Import the CWYAP_SAPAdapter_Tx.rar version of the adapter when you create a module that makes use of transactional (tRFC) or queued transactional (qRFC) processing.

Inbound processing for the ALE interfaces

The adapter supports inbound processing (from the SAP server to the adapter) for the ALE interface.

When you are configuring a module for the ALE interface, you can indicate whether the IDocs are sent as a packet. In addition, you can specify whether IDocs are sent parsed or unparsed. You make these selections on the Configuration Properties window of the J2C Bean wizard. The selections you make are reflected in the application-specific information for the IDoc business object.

The following list describes the sequence of processing actions that result from an inbound request using the ALE interface.

1. The adapter starts event listeners to the SAP server.
2. Whenever an event occurs in SAP, the event is sent to the adapter by way of the event listeners.
3. The adapter converts the event into a business object before sending it to the endpoint.

The adapter uses the event recovery mechanism to track and recover events in case of abrupt termination. The event recovery mechanism uses a data source for persisting the event state.

Not that the adapter is able to listen to and deliver events from multiple SAP systems.

The adapter is also able to deliver events to multiple endpoints. You enable delivery to multiple endpoints by configuring multiple activation specifications.

- If the endpoints subscribe to the same events from the same SAP system, all properties in the individual activation specifications must be identical.
- Endpoints that subscribe to different activation specifications receive events that match the criteria for the activation specification.

Define a separate activation specification for each endpoint to which events need to be delivered, except when the adapter delivers events only to those endpoints that are active.

Note: When multiple endpoints subscribe to the same events from the same event store, the adapter assures event delivery to active endpoints only. Any endpoints that are inactive do not receive the event. If there are multiple endpoints and any one endpoint is not active, the message is skipped for that endpoint and the adapter delivers the event only to the active endpoints. If all the endpoints are inactive, the event is rolled back, and the event needs to be resubmitted from SAP.

The following table provides an overview of the differences between the ALE interface and the ALE pass-through IDoc interface for inbound processing.

Table 2.

Interface	When to use	SplitIDoc = true	SplitIDoc = false	Parsed IDoc = true
ALE inbound	This interface converts the raw incoming IDocs to business objects, which are readily consumable by the client at the endpoint.	On receiving the IDoc packet from SAP, the adapter converts the IDocs to business objects, one by one, before sending each one to the endpoint.	On receiving the IDoc packet from SAP, the adapter converts the IDocs in the packet as one business object before sending it to the endpoint.	The incoming IDoc is only partially parsed (the control record of the IDoc is parsed but the data record is not). The client at the endpoint is responsible for parsing the data record.
ALE pass-through IDoc	This interface wraps the raw incoming IDoc in a business object before delivering it to the client at the endpoint. The client is responsible for parsing the raw IDoc.	On receiving the IDoc packet from SAP, the adapter wraps each raw IDoc within a business object before sending the objects, one by one, to the endpoint.	On receiving the IDoc packet from SAP, the adapter wraps the raw IDoc packet in a business object before sending it to the endpoint.	This attribute is not applicable to the ALE pass-through IDoc interface. (Neither the control record nor the data record of the IDoc is parsed.

Event error handling

WebSphere Adapter for SAP Software provides error handling for inbound ALE events by logging the errors and attempting to restart the event listener.

When the adapter detects an error condition, it performs the following actions:

1. The adapter logs the error information in the event log or trace file.

   Log and trace files are in the /profiles/*profile_name*/logs/*server_name* path of the folder in which WebSphere Application Server is installed.
2. The adapter attempts to restart the existing event listeners.

   The adapter uses the activation specification values for RetryLimit and RetryInterval.
   - If the SAP application is not active, the adapter attempts to restart the listeners for the number of times configured in the RetryLimit property.
   - The adapter waits for the time specified in the RetryInterval parameter before attempting to restart the event listeners.
3. If the attempt to restart the event listeners fails, the adapter performs the following actions:
   a. The adapter logs the error condition in the event log or trace file.
   b. The adapter cleans up the existing ALE event listeners.
   c. The adapter starts new event listeners.

   Note: The adapter uses the values of the RetryLimit and RetryInterval properties when starting the new event listeners.
4. If all the retry attempts fail, the adapter logs the relevant message and CEI events and stops trying to recover the ALE event listener.

   Note: You must restart the adapter application in this case

Event recovery

You can configure the adapter for ALE inbound processing so that it supports event recovery in case of abrupt termination. When event recovery is specified, the adapter persists the event state in an event recovery table that resides on a data source. Event recovery is not the default; you must specify it by enabling once-only delivery of events during adapter configuration.

Data source

Event recovery for ALE inbound processing requires that a JDBC data source be configured. You use the administrative console to configure the data source. You select a JDBC provider (for example, Derby) and then create a new data source.

Event recovery table

You can create the event recovery table manually, or you can have the adapter create the event table. The value of the EP_CreateTable configuration property determines whether the event recovery table is created automatically. The default value of this property is `True` (create the table automatically).

To create the table manually, use the information provided in the following table.

Table 3. Event recovery table fields

Table field name	Type	Description
EVNTID	VARCHAR(255)	Transaction ID for the tRFC (Transactional Remote Function Call) protocol. The tRFC protocol significantly improves the reliability of the data transfer, but it does not ensure that the order of ALE transactions specified in the application is observed. Event ordering is also affected by the number of event listeners. However, at some point all ALE transactions are transferred.
EVNTSTAT	INTEGER	Event processing status. Possible values are: • 0 (Created) • 1 (Executed) • 3 (In Progress) • -1 (Rollback)
XID	VARCHAR(255)	An XA resource keeps track of transaction IDs (XIDs) in the event recovery table. The adapter queries and updates that XID field. During recovery, WebSphere Application Server calls the resource adapter, querying it for XA resources, and then does transaction recovery on them. **Note:** The XA resource is used to enable assured once delivery. Make sure the activation specification property Assured Once Delivery is set to `true`.
BQTOTAL	INTEGER	Total number of IDocs in the packet.
BQPROC	INTEGER	Sequence number of the IDoc in the packet that the adapter is currently processing.
EVNTDATA	VARCHAR(255)	Not used.

To use event recovery for multiple endpoints, you must configure a separate event recovery table for each endpoint, although you can use the same data source (for example, Derby) to hold all the event recovery tables.

Event processing for parsed IDocs

An inbound event can contain a single IDoc or multiple IDocs, with each IDoc corresponding to a single business object. The multiple IDocs are sent by the SAP server to the adapter in the form of an IDoc packet. You can specify, during adapter configuration, whether the packet can be split into individual IDocs or whether it must be sent as one object (non-split).

Event processing begins when the SAP server sends a transaction ID to the adapter. The following sequence occurs.

1. The adapter checks the status of the event and takes one of the following actions:
   - If this is a new event, the adapter stores an EVNTID (which corresponds to the transaction ID) along with a status of `0` (Created) in the event recovery table.

- If the event status is -1 (Rollback), the adapter updates the status to 0 (Created).
- If the event status is 1 (Executed), the adapter returns an indication of success to the SAP system.

2. The SAP system sends the IDoc to the adapter.
3. The adapter converts the IDoc to a business object and sends it to the endpoint.

   Note: For single IDocs and non-split IDoc packets, the adapter can deliver objects to endpoints that support transactions as well as to endpoints that do not support transactions.

   - For endpoints that support transactions, the adapter delivers the object as part of a unique XA transaction controlled by WebSphere Application Server. When the endpoint processes the event and the transaction is committed, the status of the event is updated to 1 (Executed).

     Note: The endpoint must be configured to support XA transactions.
   - For endpoints that do not support transactions, the adapter delivers the object to the endpoint and updates the status of the event to 1 (Executed). The adapter delivers the business object without the quality of service (QOS) that guarantees once only delivery.
4. For split packets only, the adapter performs the following tasks:
   a. The adapter updates the BQTOTAL column (or table field) in the event recovery table to the number of IDocs in the packet. This number is used for audit and recovery purposes.
   b. The adapter sends the business objects to the message endpoint, one after the other, and updates the BQPROC property to the sequence number of the IDoc it is working on. The adapter delivers the objects to the appropriate endpoint as part of a unique XA transaction (a two-phase commit transaction) controlled by the application server.
   c. When the endpoint receives the event and the transaction is committed, the adapter increments the number in the BQPROC property.

      Note: The message endpoint must be configured to support XA transactions.
      If the adapter encounters an error while processing a split IDoc packet, it can behave in one of two ways, depending on the IgnoreIDocPacketErrors configuration property:

      - If the IgnoreIDocPacketErrors property is set to false, the adapter stops processing any additional IDocs in the packet and reports errors to the SAP system.
      - If the IgnoreIDocPacketErrors property is set to true, the adapter logs an error and continues processing the rest of the IDocs in the packet. The status of the transaction is marked 3 (InProgress). In this case, the adapter log shows the IDoc numbers that failed, and you must resubmit those individual IDocs separately. You must also manually maintain these records in the event recovery table.

      This property is not used for single IDocs and for non-split IDoc packets.
   d. The SAP system sends a COMMIT call to the adapter.
   e. After the adapter delivers all the business objects in the IDoc packet to the message endpoint, it updates the event status to 1 (Executed).
   f. In case of abrupt interruptions during IDoc packet processing, the adapter resumes processing the IDocs from the current sequence number. The

adapter continues updating the BQPROC property, even if IgnoreIDocPacketErrors is set to `true`. The adapter continues the processing in case you terminate the adapter manually while the adapter is processing an IDoc packet.

5. If an exception occurs either while the adapter is processing the event or if the endpoint generates an exception, the event status is updated to `-1` (Rollback).
6. If no exception occurs, the SAP server sends a CONFIRM call to the adapter.
7. The adapter deletes the records with a 1 (Executed) status and logs a common event infrastructure (CEI) event that can be used for tracking and auditing purposes.

Event processing for unparsed IDocs

Unparsed IDocs are passed through, with no conversion of the data (that is, the adapter does not parse the data part of the IDoc). The direct exchange of IDocs in the adapter enables high-performance, asynchronous interaction with SAP, because the parsing and serializing of the IDoc occurs outside the adapter. The consumer of the IDoc parses the IDoc.

The adapter processes the data based on whether the packet IDoc is split or non-split and whether the data needs to be parsed.

- The adapter can process packet IDocs as a packet or as individual IDocs. When an IDoc is received by the adapter from SAP as a packet IDoc, it is either split and processed as individual IDocs, or it is processed as a packet. The value of the SplitIDocPacket metadata at the business-object level determines how the IDoc is processed.

  In the case of split IDocs, the wrapper contains only a single, unparsed IDoc object.
- The Type metadata specifies whether the data should be parsed. For unparsed IDocs, this value is `UNPARSEDIDOC`; for parsed IDocs, the value is `IDOC`. This value is set by J2C Bean wizard.

Unparsed data format

In the fixed-width format of an unparsed IDoc, the segment data of the IDoc is set in the IDocData field of the business object. It is a byte array of fixed-length data.

The entire segment length might not be used. The adapter pads spaces to the fields that have data; the rest of the fields are ignored, and an end of segment is set. The end of segment is denoted by null.

Limitations

The unparsed event feature introduces certain limitations on the enterprise application for a particular IDoc type.

- The enterprise application supports either parsed or unparsed business-object format for a given IDoc type or message type.
- For a given IDoc type, if you select unparsed business-object format for inbound, you cannot have inbound and outbound interfaces in the same EAR file, because outbound is based on parsed business objects.
- The DummyKey feature is not supported for unparsed IDocs.

IDoc status updates

To monitor IDoc processing, you can configure the adapter to update the IDoc status. When the adapter configuration property ALEUpdateStatus is set to `true`

(indicating that an audit trail is required for all message types), the adapter updates the IDoc status of ALE business objects that are retrieved from the SAP server. After the event is sent to the message endpoint, the adapter updates the status of the IDoc in SAP to indicate whether the processing succeeded or failed. Monitoring of IDocs applies only to inbound processing (when the IDoc is sent from the SAP server to the adapter).

The adapter updates a status IDoc (ALEAUD) and sends it to the SAP server.

An IDoc that is not successfully sent to the endpoint is considered a failure, and the IDoc status is updated by the adapter. Similarly, an IDoc that reaches the endpoint is considered successfully processed, and the status of the IDoc is updated.

The status codes and their associated text are configurable properties of the adapter, as specified in the activation specification properties and shown in the following list:

- ALESuccessCode
- ALEFailureCode
- ALESuccessText
- ALEFailureText

Perform the following tasks to ensure that the adapter updates the standard SAP status code after it retrieves an IDoc:

- Set the AleUpdateStatus configuration property to `true` and set values for the AleSuccessCode and AleFailureCode configuration properties.
- Configure the inbound parameters of the partner profile of the logical system in SAP to receive the ALEAUD message type. Set the following properties to the specified values:

Table 4. Inbound properties of the logical system partner profile

SAP property	**Value**
Basic Type	ALEAUD01
Logical Message Type	ALEAUD
Function module	IDOC_INPUT_ALEAUD
Process Code	AUD1

Java data bindings for the ALE interface

A Java data binding is a structure that consists of data, the action to be performed on the data, and additional instructions for processing the data.

The WebSphere Adapter for SAP Software depends on the IDoc meta data that is generated by the J2C Bean wizard to construct Java data bindings. This metadata contains ALE-related information such as segment information, field names and an indication of whether the Java data binding handles a single IDoc or an IDoc packet.

You create Java data bindings by using J2C Bean wizard, launched from IBM Rational Application Developer for WebSphere Software. The wizard connects to the application, discovers data structures in the application, and generates Java

data bindings to represent them. It also generates other artifacts needed by the adapter such as the interface information that indicates the input and output parameters.

Java Data Binding structure for the ALE interface

During ALE processing, the adapter exchanges business objects with the SAP application. The business object represents an individual IDoc or an IDoc packet. This business object is a top-level wrapper object that contains one or more IDoc child objects, each one corresponding to a single IDoc. The same business object format is used for inbound and outbound processing.

Wrapper business object

The wrapper business object contains a transaction ID, a queue name, and one or more IDoc business objects. The transaction ID (SAPTransactionID) is used to ensure once-only delivery of business objects, and the queue name (qRFCQueueName) specifies the name of the queue on the SAP server to which the IDocs should be delivered. If you are not using transaction IDs or queues, these properties are blank.

For individual IDocs, the wrapper business object contains only one instance of an IDoc business object. For IDoc packets, the wrapper business object contains multiple instances of an IDoc business object.

Note that the transaction ID and queue name attributes are present in the business object even if you are not using the tRFC or qRFC features.

IDoc business object

The IDoc business object (SapAlereq01IDocBO, in the example) has the following structure:The IDoc business object contains the following objects.

Control record

The control record business object contains the metadata required by the adapter to process the business object.

Data record

The data record business object contains the actual business object data to be processed by the SAP application and the metadata required for the adapter to convert it to an IDoc structure for the RFC call.

Unparsed IDocs

For an unparsed IDoc, in which the data part of the IDoc is not parsed by the adapter, the IDoc business object contains a dummy key, a control record, and the IDoc data.

Application-specific information

Additional information about the business object can be found in the application-specific information of the business object. For example, the application-specific information for SapAleReq01 lists whether the IDoc packet is split and provides information about the operation.

Dummy keys

You use a dummy key to map a key field from an IDoc control or data record business object to the dummyKey property of the top-level business object. The dummyKey property is used for flow control and business process logic. You can use the dummyKey when you need the top-level business object to participate in a relationship.

The adapter supports dummy key mapping in the following manner:

- You must configure the property-level application-specific information of the dummyKey property as the path to the property from which the value should be set. For example: dataRecord/SapOrders05e2edk01005/idocDocumentNumber
- Multiple cardinality objects are not supported. If the path contains a multiple cardinality object, the value is ignored and the default first index is used.
- If the application-specific information is incorrect or if the mapped property value is empty, the adapter causes the event to fail. This is also the case when the application-specific information is configured to set an object type value as the dummyKey.

  Note: The dummyKey property can contain only a simple type.

Dummy key processing is not supported for unparsed IDocs.

The ALE pass-through IDoc interface

The ALE pass-through IDoc interface enables business process integration and asynchronous data communication between two or more SAP systems or between SAP and external systems. The data is exchanged in the form of Intermediate Documents (IDocs).

The adapter supports outbound and inbound processing by enabling the exchange of data in the form of business objects.

- For inbound processing, SAP pushes the data in IDocs to the SAP adapter. The adapter converts the IDocs to business objects and delivers them to the endpoint.
- For outbound processing, the SAP adapter converts the business object to an IDoc and delivers it to SAP.

To use the ALE pass-through IDoc interface for inbound processing, you must make sure that your SAP server is properly configured (for example, you must set up a partner profile and register a program ID to listen for events).

Application systems are loosely coupled in an ALE integrated system, and the data is exchanged asynchronously.

IDocs

IDocs are containers for exchanging data in a predefined (structured ASCII) format across system boundaries. The IDoc type indicates the SAP format that is to be used to transfer the data. An IDoc type can transfer several message types (the logical messages that correspond to different business processes). IDocs are used for outbound and inbound processing.

IDocs are exchanged for inbound and outbound events, and IDocs can be exchanged either as individual documents or in packets. For both outbound and

inbound processing, the adapter does no conversion of the IDoc. This is useful when the client wants to perform the IDoc parsing.

Transactional RFC processing

The adapter uses tRFC (transactional RFC) to guarantee delivery and to ensure that each IDoc is exchanged only once with SAP. The tRFC component stores the called RFC function in the database of the SAP system along with a unique transaction identifier (TID).

The most common reason for using transaction ID support is to ensure once and only once delivery of data. To make sure of this feature, select the transaction RAR file (CWYAP_SAPAdapter_Tx.rar) when you configure the adapter.

Note: The SAP transaction ID property is always generated by the J2C Bean wizard; however, it is supported only for outbound operations when the CWYAP_SAPAdapter_Tx.rar version of the adapter is used.

The client application must determine how to store the SAP transaction ID and how to relate the SAP transaction ID to the data being sent to the adapter. When the events are successful, the client application should not resubmit the event associated with this TID again to prevent the processing of duplicate events.

- If the client application does not send an SAP transaction ID with the business object, the adapter returns one after executing the transaction.
- If the client application has an SAP transaction ID, it needs to populate the SAP transaction ID property with that value before executing the transaction.

The SAP transaction ID can be used for cross-referencing with a global unique ID that is created for an outbound event. The global unique ID is something you can create for managing integration scenarios.

Queued RFC processing

The adapter uses qRFC (queued transactional RFC) to ensure that IDocs are delivered in sequence to a queue on the SAP server or are received in sequence from the SAP server.

Outbound processing for the ALE pass-through IDoc interface

The adapter supports outbound processing (from the adapter to the SAP server) for the ALE pass-through IDoc interface. ALE uses IDocs for data exchange, and the adapter uses business objects to represent the IDocs.

The following list describes the sequence of processing actions that result from an outbound request using ALE pass-through IDoc interface.

Note: The client application that makes the request uses the interface information that was generated by the J2C Bean wizard.

1. The adapter receives a request, which includes a wrapper business object, from a client application.

   Note: The wrapper business object contains a data stream representing the IDoc. No separate IDoc business object exists for pass-through IDocs
2. The adapter uses the IDoc wrapper business object to populate the appropriate RFC-enabled function call used by the ALE interface.

3. The adapter establishes an RFC connection to the ALE interface and passes the IDoc data to the SAP system. If you are using the qRFC protocol, the adapter passes the IDoc data in the order specified in the wrapper business object to the specified queue on the SAP server.
4. After passing the data to SAP, the adapter performs one of the following steps:
   - If the call is not managed by a J2C local transaction, the adapter releases the connection to SAP and does not return any data to the caller. When no exceptions are raised, the outbound transaction is considered successful. You can verify whether the data is incorporated into the SAP application by inspecting the IDocs that have been generated in SAP.
   - If the call is managed by a J2C local transaction, the adapter returns the transaction ID.

     The adapter uses the tRFC protocol to support J2C local transactions.

Import the CWYAP_SAPAdapter_Tx.rar version of the adapter when you create a module that makes use of transactional (tRFC) or queued transactional (qRFC) processing.

Inbound processing for the ALE pass-through IDoc interface

The adapter supports inbound processing (from the SAP server to the adapter) for the the ALE pass-through IDoc interface.

When you are configuring a module for the ALE pass-through interface, you can indicate whether the IDocs are sent as a packet. You make this selection on the Configuration Properties window of the J2C Bean wizard. The selection you make is reflected in the application-specific information for the IDoc wrapper business object.

Note: When you use the ALE pass-through IDoc interface, a wrapper business object contains a data stream representing the IDoc. No separate IDoc business object exists for pass-through IDocs.

The following list describes the sequence of processing actions that result from an inbound request using the ALE interface.
1. The adapter starts event listeners to the SAP server.
2. Whenever an event occurs in SAP, the event is sent to the adapter by way of the event listeners.
3. The adapter converts the event into a business object before sending it to the endpoint.

The adapter uses the event recovery mechanism to track and recover events in case of abrupt termination. The event recovery mechanism uses a data source for persisting the event state.

Not that the adapter is able to listen to and deliver events from multiple SAP systems.

The adapter is also able to deliver events to multiple endpoints. You enable delivery to multiple endpoints by configuring multiple activation specifications that exist in the same module.

- If the endpoints subscribe to the same events from the same SAP system, all properties in the individual activation specifications must be identical.
- Endpoints that subscribe to different activation specifications receive events that match the criteria for the activation specification.

Define a separate activation specification for each endpoint to which events need to be delivered, except when the adapter delivers events only to those endpoints that are active.

Note: When multiple endpoints subscribe to the same events from the same event store, the adapter assures event delivery to active endpoints only. Any endpoints that are inactive do not receive the event. If there are multiple endpoints and any one endpoint is not active, the message is skipped for that endpoint and the adapter delivers the event only to the active endpoints. If all the endpoints are inactive, the event is rolled back, and the event needs to be resubmitted from SAP.

The following table provides an overview of the differences between the ALE interface and the ALE pass-through IDoc interface for inbound processing.

Table 5.

Interface	When to use	SplitIDoc = true	SplitIDoc = false	Parsed IDoc = true
ALE inbound	This interface converts the raw incoming IDocs to business objects, which are readily consumable by the client at the endpoint.	On receiving the IDoc packet from SAP, the adapter converts the IDocs to business objects, one by one, before sending each one to the endpoint.	On receiving the IDoc packet from SAP, the adapter converts the IDocs in the packet as one business object before sending it to the endpoint.	The incoming IDoc is only partially parsed (the control record of the IDoc is parsed but the data record is not). The client at the endpoint is responsible for parsing the data record.
ALE pass-through IDoc	This interface wraps the raw incoming IDoc in a business object before delivering it to the client at the endpoint. The client is responsible for parsing the raw IDoc.	On receiving the IDoc packet from SAP, the adapter wraps each raw IDoc within a business object before sending the objects, one by one, to the endpoint.	On receiving the IDoc packet from SAP, the adapter wraps the raw IDoc packet in a business object before sending it to the endpoint.	This attribute is not applicable to the ALE pass-through IDoc interface. (Neither the control record nor the data record of the IDoc is parsed.

Event error handling

WebSphere Adapter for SAP Software provides error handling for inbound ALE events by logging the errors and attempting to restart the event listener.

When the adapter detects an error condition, it performs the following actions:

1. The adapter logs the error information in the event log or trace file.

   Log and trace files are in the /profiles/*profile_name*/logs/*server_name* path of the folder in which WebSphere Application Server is installed.
2. The adapter attempts to restart the existing event listeners.

   The adapter uses the activation specification values for RetryLimit and RetryInterval.

- If the SAP application is not active, the adapter attempts to restart the listeners for the number of times configured in the RetryLimit property.
- The adapter waits for the time specified in the RetryInterval parameter before attempting to restart the event listeners.

3. If the attempt to restart the event listeners fails, the adapter performs the following actions:
   a. The adapter logs the error condition in the event log or trace file.
   b. The adapter cleans up the existing ALE event listeners.
   c. The adapter starts new event listeners.

   Note: The adapter uses the values of the RetryLimit and RetryInterval properties when starting the new event listeners.
4. If all the retry attempts fail, the adapter logs the relevant message and CEI events and stops trying to recover the ALE event listener.

   Note: You must restart the adapter application in this case

Event recovery

You can configure the adapter for ALE inbound processing so that it supports event recovery in case of abrupt termination. When event recovery is specified, the adapter persists the event state in an event recovery table that resides on a data source. Event recovery is not the default; you must specify it by enabling once-only delivery of events during adapter configuration.

Data source

Event recovery for ALE inbound processing requires that a JDBC data source be configured. You use the administrative console to configure the data source. You select a JDBC provider (for example, Derby) and then create a new data source.

Event recovery table

You can create the event recovery table manually, or you can have the adapter create the event table. The value of the EP_CreateTable configuration property determines whether the event recovery table is created automatically. The default value of this property is `True` (create the table automatically).

To create the table manually, use the information provided in the following table.

Table 6. Event recovery table fields

Table field name	Type	Description
EVNTID	VARCHAR(255)	Transaction ID for the tRFC (Transactional Remote Function Call) protocol. The tRFC protocol significantly improves the reliability of the data transfer, but it does not ensure that the order of ALE transactions specified in the application is observed. Event ordering is also affected by the number of event listeners. However, at some point all ALE transactions are transferred.

Table 6. Event recovery table fields (continued)

Table field name	Type	Description
EVNTSTAT	INTEGER	Event processing status. Possible values are: • 0 (Created) • 1 (Executed) • 3 (In Progress) • -1 (Rollback)
XID	VARCHAR(255)	An XA resource keeps track of transaction IDs (XIDs) in the event recovery table. The adapter queries and updates that XID field. During recovery, WebSphere Application Server calls the resource adapter, querying it for XA resources, and then does transaction recovery on them. **Note:** The XA resource is used to enable assured once delivery. Make sure the activation specification property Assured Once Delivery is set to `true`.
BQTOTAL	INTEGER	Total number of IDocs in the packet.
BQPROC	INTEGER	Sequence number of the IDoc in the packet that the adapter is currently processing.
EVNTDATA	VARCHAR(255)	Not used.

To use event recovery for multiple endpoints, you must configure a separate event recovery table for each endpoint, although you can use the same data source (for example, Derby) to hold all the event recovery tables.

IDoc status updates

To monitor IDoc processing, you can configure the adapter to update the IDoc status. When the adapter configuration property ALEUpdateStatus is set to `true` (indicating that an audit trail is required for all message types), the adapter updates the IDoc status of ALE business objects that are retrieved from the SAP server. After the event is sent to the message endpoint, the adapter updates the status of the IDoc in SAP to indicate whether the processing succeeded or failed. Monitoring of IDocs applies only to inbound processing (when the IDoc is sent from the SAP server to the adapter).

The adapter updates a status IDoc (ALEAUD) and sends it to the SAP server.

An IDoc that is not successfully sent to the endpoint is considered a failure, and the IDoc status is updated by the adapter. Similarly, an IDoc that reaches the endpoint is considered successfully processed, and the status of the IDoc is updated.

The status codes and their associated text are configurable properties of the adapter, as specified in the activation specification properties and shown in the following list:

- ALESuccessCode
- ALEFailureCode
- ALESuccessText
- ALEFailureText

Perform the following tasks to ensure that the adapter updates the standard SAP status code after it retrieves an IDoc:

- Set the AleUpdateStatus configuration property to `true` and set values for the AleSuccessCode and AleFailureCode configuration properties.
- Configure the inbound parameters of the partner profile of the logical system in SAP to receive the ALEAUD message type. Set the following properties to the specified values:

Table 7. Inbound properties of the logical system partner profile

SAP property	Value
Basic Type	ALEAUD01
Logical Message Type	ALEAUD
Function module	IDOC_INPUT_ALEAUD
Process Code	AUD1

ALE pass-through IDoc business object structure

During ALE processing, the adapter exchanges business objects with the SAP application. The business object represents an individual IDoc or an IDoc packet. For pass-through IDocs, the business object contains an IDoc stream instead of a child business object. The same business object format is used for inbound and outbound ALE pass-through IDoc processing.

The business object contains a transaction ID, a queue name, stream data, and the IDoc type. The transaction ID (SAPTransactionID) is used to ensure once-only delivery of business objects, and the queue name (qRFCQueueName) specifies the name of the queue on the SAP server to which the IDocs should be delivered. If you are not using transaction IDs or queues, these properties are blank.

Additional information about the business object can be found in the application-specific information of the business object. For example, the application-specific information for SapAleReq01 lists whether the IDoc packet is split and provides information about the type of object, which, for pass-through IDoc business objects is always PASSTHROUGHIDOC.

Query interface for SAP Software

The Query interface for SAP Software provides you with the means to retrieve data from application tables on an SAP server or to query SAP application tables for the existence of data. The adapter can perform hierarchical data retrieval from the SAP application tables.

Query interface for SAP Software supports outbound interactions for read operations (RetrieveAll and Exists) only. You can use this interface in local transactions to look up records before write operations (Create, Update, or Delete). For example, you can use the interface as part of a local transaction to do an existence check on a customer before creating a sales order. You can also use the interface in non-transaction scenarios.

Query interface for SAP Software supports data retrieval form SAP application tables, including hierarchical data retrieval from multiple tables. The interface supports static as well as dynamic specification of `where` clauses for the queries.

The J2C Bean wizard finds the application data tables in SAP, interprets the hierarchical relationship between tables, and constructs a representation of the tables and their relationship in the form of a business object. The wizard also builds a default `where` clause for the query.

You can control the depth of the data retrieval as well as the amount of information using the maxRow and rowsSkip properties.

Query interface for SAP software (QISS) supports hierarchical data retrieval from the SAP application tables. Using this interface, the adapter can determine the existence of data in SAP application tables or retrieve all the data in SAP application tables. For example, if the client wants to check if customer Bob exists in SAP system, he/she would run the external service wizard to discover the SAP application table KNA1. The wizard then generates the business object for KNA1 along with other artifacts. At runtime, the client passes the KNA1 business object to the adapter to invoke the QISS interface, the adapter retrieves the table data from SAP and returns the result to the calling client.

Outbound processing for the query interface for SAP Software

You use the Query interface for SAP Software for outbound processing only.

Note: The client application that makes the request uses the interface information that was generated by the J2C Bean wizard.

The following list describes the sequence of processing actions that result from an outbound request using the query interface for SAP Software.

1. The adapter receives a request, which includes a table object, from a client application.
2. The adapter determines, from the table object sent with the query, the name of the table to examine.
3. The adapter determines the columns to retrieve or examine.
4. The adapter determines the rows to retrieve or examine.
5. The adapter responds.
   - In the case of a RetreiveAll operation, the adapter returns a result set in the form of a container of query business objects, which represent the data for each row retrieved from the table. If the query is received as a table business object (not inside a container), the rows are returned one at a time, as they are retrieved.
   - In the case of the Exists operation, the adapter returns an indication of whether the data exists in the SAP table.
   - If no data exists, the adapter generates an exception.

Business objects for the query interface for SAP Software

The input to the Query interface for SAP Software is a table business object. The table business object represents the columns in a table on the SAP server. The adapter uses the table business object to obtain data from tables on the SAP server.

Business object structure

The table business object can be part of a container.

The table business object contains columns selected from the specified SAP table.

In addition to column information, the table business object also contains a query business object as the last parameter.

The properties of the query business object are sapWhereClause, sapRowsSkip, and sapMaxRows:

- The sapWhereClause property retrieves information from SAP tables. The default value is populated by the J2C Bean wizard. The space character is used as the delimiter to parse the sapWhereClause.
- The sapMaxRows property is the maximum number of rows to be returned. The default value is `100`.
- The sapRowsSkip property is the number of rows to skip before retrieving data. The default value is `0`.

The tables can be modeled as hierarchical business objects. You specify the parent-child relationship of the tables in the J2C Bean wizard.

Tables are linked by a foreign key to form parent-child relationships. The child table business object has a foreign key that references a property in the parent query business object.

The return from the Query interface for SAP Software call for a RetrieveAll operation is a container of business graphs or a container of table objects.

The Advanced event processing interface

The Advanced event processing interface of the WebSphere Adapter for SAP Software is used for both inbound and outbound processing. For inbound processing, it polls for events in SAP, converts them into business objects, and sends the event data as business objects to WebSphere Application Server. For outbound processing, the adapter processes events sent from an application to retrieve data from or update data in the SAP server.

You can use the WebSphere BI Station tool to monitor inbound events.

Advanced event processing interface supports both inbound and outbound processing. For inbound, the adapter polls the SAP system for events and delivers the events to the endpoint. For this interface, the user needs to write custom ABAP handler on the SAP system. The ABAP handler is invoked by the adapter at runtime. This is the most complex interface to use. The application developer would use this interface if the other interfaces cannot provide the capability needed for the application that is being developed.

Note: You must select the non-transaction RAR, which is CWYAP_SAPAdapter.rar, when you configure the adapter to make use of the Advanced event processing interface.

Outbound processing for the Advanced event processing interface

During outbound processing, business object data is converted into an ABAP handler function, which is called on the SAP server. When the data is returned by the ABAP handler function, the data is converted to a business object, and the business object is returned as a response.

The following list describes the sequence of processing actions that result from an outbound request using the Advanced event processing interface.

1. The adapter receives the Advanced event processing business object, which contains business data along with the metadata.
2. The Advanced event processing interface of the adapter uses the metadata of the business object to obtain the type of IDoc specified and to reformat the business object data into the structure of that IDoc.
3. After it reformats the data, the adapter passes the business object data to an object-specific ABAP handler (based on the operation), which handles the integration with an SAP native API.
4. After the object-specific ABAP handler finishes processing the business object data, it returns the response data in IDoc format to the adapter, which converts it to the business object.
5. The adapter returns the results to the caller.

ABAP handler overview

An ABAP handler is a function module that gets data into and out of the SAP application database. For each business object definition that you develop, you must support it by developing a custom ABAP handler.

ABAP handlers reside in the SAP application as ABAP function modules. ABAP handlers are responsible for adding business-object data into the SAP application database (for Create, Update, and Delete operations) or for using the business-object data as the keys to retrieving data from the SAP application database (for the Retrieve operation).

You must develop operation-specific ABAP handlers for each hierarchical business object that needs to be supported. If you change the business-object definition, you must also change the ABAP handler.

The ABAP handler can use any of the SAP native APIs for handling the data. Some of the native APIs are listed below.

- Call Transaction

  Call Transaction is the SAP-provided functionality for entering data into an SAP system. Call Transaction guarantees that the data adheres to the SAP data model by using the same screens an online user sees in a transaction. This process is commonly referred to as *screen scraping*.
- Batch data communication (BDC)

  Batch Data Communication (BDC) is an instruction set that SAP can follow to process a transaction without user intervention. The instructions specify the sequence in which the screens in a transaction are processed and which fields are populated with data on which screens. All of the elements of an SAP transaction that are exposed to an online user have identifications that can be used in a BDC.
- ABAP SQL

  ABAP SQL is the SAP proprietary version of SQL. It is database- and platform-independent, so that whatever SQL code you write, you can run it on any database and platform combination that SAP supports. ABAP SQL is similar in syntax to other versions of SQL and supports all of the basic database table commands such as update, insert, modify, select, and delete. For a complete description of ABAP SQL, see your SAP documentation.

Using ABAP SQL, an ABAP handler can modify SAP database tables with business object data for create, update, and delete operations. It can also use the business object data in the where clause of an ABAP select statement as the keys.

Note: Use of ABAP SQL to modify SAP tables is not recommended, because it might corrupt the integrity of the database. Use ABAP SQL only to retrieve data.

- ABAP Function Modules and Subroutines

  From the ABAP handler, you can call ABAP function modules or subroutines that implement the required function.

The adapter provides the following tools to help in the development process:

- The adapter includes the Call Transaction Recorder Wizard to assist you in developing the ABAP handlers that use call transactions or BDC sessions.
- The J2C Bean wizard generates the required business objects and other artifacts for Advanced event processing. The business objects are based on IDocs, which can be custom or standard.
- The adapter provides samples that you can refer to for an understanding of the Advanced event processing implementation.

ABAP handler creation

For each IDoc object definition that you develop, you must support it by developing a custom ABAP handler.

You can use either standard IDocs or custom IDocs for the Advanced event processing interface. After defining the custom IDoc for an integration scenario, create an ABAP handler (function module) for each operation of the business object that needs to be supported.

Each function should have the following interface to ensure that the adapter can call it:

```
*" IMPORTING
*" VALUE(OBJECT_KEY_IN) LIKE /CWLD/LOG_HEADER-OBJ_KEY OPTIONAL
*" VALUE(INPUT_METHOD) LIKE BDWFAP_PAR-INPUTMETHD OPTIONAL
*" VALUE(LOG_NUMBER) LIKE /CWLD/LOG_HEADER-LOG_NR OPTIONAL
*" EXPORTING
*" VALUE(OBJECT_KEY_OUT) LIKE /CWLD/LOG_HEADER-OBJ_KEY
*" VALUE(RETURN_CODE) LIKE /CWLD/RFCRC_STRU-RFCRC
*" VALUE(RETURN_TEXT) LIKE /CWLD/LOG_HEADER-OBJ_KEY
*" TABLES
*" IDOC_DATA STRUCTURE EDID4
*" LOG_INFO STRUCTURE /CWLD/EVENT_INFO
```

The following table provides information about the parameters:

Table 8. Interface parameters

Parameter	Description
OBJECT_KEY_IN	Should be no value.

Table 8. Interface parameters (continued)

Parameter	Description
INPUT_METHOD	Indicates whether the IDoc should be processed in a dialog (that is, through Call Transaction). Possible values are: ″ ″ - Background (no dialog) ″A″ - Show all screens ″E″ - Start the dialog on the screen where the error occurred “N” Default
LOG_NUMBER	Log Number.
OBJECT_KEY_OUT	Customer ID returned from the calling transaction.
RETURN_CODE	0 - Successful. 1 - Failed to retrieve. 2 - Failed to create, update, or delete.
RETURN_TEXT	Message describing the return code.
IDOC_DATA	Table containing one entry for each IDoc data segment. The following fields are relevant to the inbound function module: Docnum - The IDoc number. Segnam - The segment name. Sdata - The segment data.
LOG_INFO	Table containing details regarding events processed with either a success or error message.

Call Transaction Recorder wizard

The adapter provides the Call Transaction Recorder Wizard to assist you in developing the ABAP handlers that use call transactions or BDC sessions.

The Call Transaction Recorder wizard enables you to generate sample code for call transactions to facilitate development. It generates sample code stubs for each screen that is modified during the recording phase.

To access this wizard, enter the /CWLD/HOME_AEP transaction in the SAP GUI.

The following is sample code generated by the wizard. You can adopt this code in the ABAP Handler.

```
* Customer master: request screen chnge/displ cent.
perform dynpro_new using 'SAPMF02D' '0101' .

* Customer account number
perform dynpro_set using 'RF02D-KUNNR' '1' .

* Function Command
perform dynpro_set using 'BDC_OKCODE' '/00' .

* Function Command
perform dynpro_set using 'BDC_OKCODE' '/00' .

* Customer master: General data, CAM address, communication
perform dynpro_new using 'SAPMF02D' '0111' .
```

```
* Title
perform dynpro_set using 'SZA1_D0100-TITLE_MEDI' 'Mr.' .

* Function Command
perform dynpro_set using 'BDC_OKCODE' '=UPDA' .

* Call Transaction
Call Transaction 'XD02' using bdcdata
    mode input_mode
   update 'S'
  messages into bdc_messages.
```

The wizard does not generate the required business object. You use the J2C Bean wizard to generate the business object.

Inbound processing for the Advanced event processing interface

The adapter uses the Advanced event processing interface to poll for events on the SAP server, to process the events, and to send them to an endpoint.

The following list describes the sequence of processing actions that result from an inbound request using the Advanced event processing interface.

1. A triggered event enters the event table with an initial status of pre-queued.
2. When the adapter polls for events, the status of the event changes from pre-queued to queued if there are no database locks for the combination of the user who created the event and the event key.
3. After the event is retrieved from the event table, the status of the event is updated to InProgress.

   If locks exist, the status of the event is set to locked and the event is re-queued into the queue. Every event with a pre-queued or locked status is updated with every poll. You can configure the polling frequency using the Poll Frequency property.
4. After preprocessing all pre-queued events, the adapter selects the events.

   The property Poll Quantity determines the maximum number of events returned for a single poll call.
5. For each event, the adapter uses the remote function specified for the Retrieve operation to retrieve the data and send it to the endpoint.

   If the AssuredOnceDelivery property is set to true, an XID value is set for each event in the event store. After each event is picked up for processing, the XID value for that event is updated in the event table.

   If before the event is delivered to the endpoint, the SAP connection is lost or the application is stopped, and the event is consequently not processed completely, the XID column ensures that the event is reprocessed and sent to the endpoint. After the SAP connection is reestablished or the adapter starts up again, it first checks for events in the event table that have a value in the XID column. It then processes these events first and then polls the other events during the poll cycles.
6. After each event is processed, it is updated or archived in the SAP application.

   When the event is processed successfully, it is archived and then deleted from the event table.

   The adapter can also filter the events to be processed by business object type. The filter is set in the Event Filter Type property. This property has a comma-delimited list of business object types, and only the types specified in

the property are picked for processing. If no value is specified for the property, no filter is applied and all the events are picked up for processing.

Event detection

Event detection refers to the collection of processes that notify the adapter of SAP application object events. Notification includes, but is not limited to, the type of the event (object and operation) and the data key required for the external system to retrieve the associated data.

Event detection is the process of identifying that an event was generated in the SAP application. Typically, adapters use database triggers to detect an event. However, because the SAP application is tightly integrated with the SAP database, SAP allows very limited access for direct modifications to its database. Therefore, the event-detection mechanisms are implemented in the application transaction layer above the database.

Adapter-supported event detection mechanisms

The four event-detection mechanisms supported by the adapter are described in the following list:

- Custom Triggers, which are implemented for a business process (normally a single SAP transaction) by inserting event detection code at an appropriate point within the SAP transaction
- Batch programs, which involve developing an ABAP program containing the criteria for detecting an event
- Business workflows, which use the object-oriented event detection capabilities of SAP
- Change pointers, a variation of business workflows, which use the concept of change documents to detect changes for a business process

All these event-detection mechanisms support real-time triggering and retrieval of objects. In addition, custom triggers and batch programs provide the ability to delay the retrieval of events. An event whose retrieval is delayed is called a future event.

Note: Each event detection mechanism has advantages and disadvantages that need to be considered when designing and developing a business object trigger. Keep in mind that these are only a few examples of event detection mechanisms. There are many different ways to detect events.

After you determine the business process to support (for example, sales quotes or sales orders) and determine the preferred event-detection mechanism, implement the mechanism for your business process.

Note: When implementing an event detection mechanism, it is a good idea to support all of the functionality for a business process in one mechanism. This limits the impact in the SAP application and makes event detection easier to manage.

See the related topics on implementing the event-detection mechanisms in the *Performing prerequisite tasks specific to an interface* section.

Event table

Events that are detected are stored in an SAP application table. This event table is delivered as part of the ABAP component. The event table structure is as follows.

Table 9. Event table fields

Name	Type	Description
event_id	NUMBER	Unique event ID that is a primary key for the table.
object_name	STRING	business object name.
object_key	STRING	Delimited string that contains the keys for the business object.
object_function	STRING	Operation corresponding to the event (Delete, Create, or Update).
event_priority	NUMBER	Any positive integer to denote the priority of the event.
event_time	DATE	Date and time when the event was generated.
event_status	NUMBER	Event processing status. Possible values are: 0 - Ready for poll 1 - Event delivered 2 - Event prequeued 3 - Event in progress 4 - Event locked -1 - Event failed
Xid	STRING	Unique XID (transaction ID) value for assured-once delivery.
event_user	STRING	User who created the event.
event_comment	STRING	Description of the event.

Event triggers

After an event is identified by one of the event-detection mechanisms, it is triggered by one of the adapter-delivered event triggers. Event triggers can cause events to be processed immediately or in the future.

The function modules that trigger events are described in the following list.

- /CWLD/ADD_TO_QUEUE_AEP

  This function module triggers events to the current event table for immediate processing.
- /CWLD/ADD_TO_QUEUE_IN_FUTURE_AEP

  This function module triggers events to the future event table to be processed at a later time.

Note: Both functions are for real-time triggering.

Current® event table

If the event will be triggered in real-time, /CWLD/ADD_TO_QUEUE_AEP commits the event to the current event table (/CWLD/EVT_CUR_AEP). Specifically, it adds a row of data for the object name, verb, and key that represents the event.

Future event table

If an event needs to be processed at a future date, the processing described in the following list.

1. A custom ABAP handler calls /CWLD/ADD_TO_QUEUE_IN_FUTURE_AEP with the event.
2. The /CWLD/ADD_TO_QUEUE_IN_FUTURE_AEP module commits the event to the future event table (/CWLD/EVT_FUT_AEP). Specifically, it adds a row of data for the object name, verb, and key that represents the event. In addition, it adds a Date row
3. The adapter-delivered batch program /CWLD/SUBMIT_FUTURE_EVENTS_AEP reads the future event table.
4. If scheduled to do so, the batch program retrieves events from the future event table.
5. After it retrieves an event, the batch program calls /CWLD/ADD_TO_QUEUE_AEP.
6. The /CWLD/ADD_TO_QUEUE_AEP module triggers the event to the current event table.

/CWLD/ADD_TO_QUEUE_IN_FUTURE_AEP uses the system date as the current date when it populates the Date row of the future event table.

Event restriction

Use event restriction to filter out events that you do not want added to the event table. The adapter provides an ABAP include program (TRIGGERING_RESTRICTIONS_USER) that can be modified to filter events.

The, TRIGGERING_RESTRICTIONS_USER program is called from within the event trigger /CWLD/ADD_TO_QUEUE_AEP to enable additional filtering of events.

Note: You must have developer privileges to make changes because the code needs to be recompiled.

To view or modify the include program TRIGGERING_RESTRICTIONS_USER:

1. If IBM WebSphere BI Station is not currently displayed, enter transaction `/n/CWLD/HOME_AEP`.
2. Click the **Configuration** tab.
3. Click **Event Restriction**.

To upgrade an adapter-provided ABAP handler from one SAP R/3 version to another, check to see if changes were made to program TRIGGERING_RESTRICTIONS_USER. This program is intended for customer modification. If changes were made, you can avoid conflicts by downloading the custom work as text files, not as transport files, for use as a reference.

Upgrade any ABAP code from the old event restriction program to the new event restriction program.

Business objects for the Advanced event processing interface

During Advanced event processing, the adapter exchanges business objects with the SAP application. The business object represents a custom IDoc, or a standard or extension IDoc available on the SAP server

Business object structure

Note: For custom interfaces that you want to support, as a first step, you need to define the custom IDoc in the SAP system. You can then use the J2C Bean wizard to discover this custom IDoc and build the required artifacts, including the business-object definition.

Note that the transaction ID and queue name attributes are present in the business object even if you are not using the tRFC or qRFC features.

The IDoc business object contains the following objects:

- The control record business object contains the metadata required by the adapter to process the business object.
- The data record business object contains the actual business object data to be processed by the SAP application and the metadata required for the adapter to convert it to an IDoc structure for the RFC call.
- The business object data (which is pointed to from the data record) has the following structure:

Additional information about the business object can be found in the application-specific information of the business object. For example, the application-specific information lists whether the IDoc packet is split and provides information about the operation.

Chapter 4. Configuring the module for deployment

To configure the adapter so that it can be deployed on WebSphere Application Server, use Rational Application Developer for WebSphere Software to create a module, which is exported as an EAR file when you deploy the adapter. You then specify the business objects you want to discover and the system on which you want to discover them.

Configuration on the SAP system

Depending on the interface you will be using, you might have to perform some prerequisite tasks before you use the J2C Bean wizard to configure the module. For example, if you are configuring a module for ALE or BAPI inbound processing, you must register a program ID with the SAP server. If you are going to use the Advanced event processing interface, you must install transport files on the SAP server.

Configuring the SAP system for ALE or BAPI inbound processing

Before you configure WebSphere Adapter for SAP Software for ALE inbound processing or for BAPI inbound processing, you must register an RFC destination on the SAP server. For ALE processing, you must also configure a receiver port, logical system, distribution model, and partner profile on the SAP server. See your system administrator if you are not sure whether these items have been configured.

About this task

Perform the following steps on the SAP server using the SAP GUI. Note that only the first task is required for BAPI inbound processing.

Procedure

1. Register an RFC program ID:
   a. Open transaction **SM59** (Display and Maintain RFC Destinations).
   b. Click **Create**.
   c. Type a name for the RFC destination.
   d. In the **Connection Type** field, select **T**.
   e. In the **Activation Type** field, select **Registered Server Program**.
   f. Type a Program ID.

      You will use this program ID when you configure the adapter. This value indicates to the SAP gateway which RFC-enabled functions the program ID listens for.
   g. Save your entry.
2. Set up a receiver port (for ALE processing only):
   a. Open transaction **WE21** (Ports in IDoc processing).
   b. Select **Transactional RFC**, click **Ports**, and click the Create icon.
   c. Type a name for the port and select **OK**.
   d. Type the name of the destination you created in the previous task (or select it from the list).

e. Save your entry.

3. Specify a logical system (for ALE processing only):
   a. Open transaction **BD54** (Change View Logical Systems).
   b. Click **New Entries**.
   c. Type a name for the logical system and click the Save icon.
   d. If you see the Prompts for Workbench request, click the New Request icon. Then enter a short description and click the Save icon.
   e. Click the Continue icon.
4. Configure a distribution model (for ALE processing only):
   a. Open transaction **BD64** (Maintenance of Distribution Model).
   b. Click **Distribution Model → Switch processing model**.
   c. Click **Create model view**.
   d. Type a name for the model view and click the Continue icon.
   e. Select the distribution model you created, and click **Add message type**.
   f. For outbound processing, type the logical system name you created in the previous task as **Sender** and the logical name of the SAP server as **Receiver**. Then select a message type (for example, **MATMAS**) and click the Continue icon.
   g. Select the distribution model again and click **Add message type**.
   h. For inbound processing, type the logical name of the SAP server as **Sender** and the logical system name you created in the previous task as **Receiver**. Then select a message type (for example, **MATMAS**) and click the Continue icon.
   i. Save your entry.
5. Set up a partner profile (for ALE processing only):
   a. Open transaction **WE20** (Partner Profiles).
   b. Click the Create icon.
   c. Type the name of the logical system you created in the earlier task and, for **Partner Type**, select **LS**.
   d. For **Post Processing: permitted agent**, type `US` and your user ID.
   e. Click the Save icon.
   f. In the Outbound parameters section, click the Create outbound parameter icon.
   g. In the Outbound parameters window, type a message type (for example, `MATMAS05`), select the receiver port you created in the earlier task, and select **Transfer IDoc immed**.
   h. Click the Save icon.
   i. Press F3 to return to the Partner Profiles view.
   j. In the Inbound parameters section, click the Create inbound parameter icon.
   k. In the Inbound parameters window, type a message type (for example, `MATMAS`), and a process code (for example, `MATM`).
   l. Click the Save icon.
   m. Press F3 to return to the Partner Profiles view.
   n. In the Inbound parameters section, click the Create inbound parameter icon.
   o. In the Inbound parameters window, type the following values: `ALEAUD` for **Message Type**, and `AUD1` for **Process Code**.
   p. Click the Save icon.

q. Press F3 to return to the Partner Profiles view.
r. Click the Save icon.

Results

You have performed the tasks (on the SAP server) required to use the BAPI inbound interface or the ALE interface.

What to do next

Configure the adapter for the interface.

Creating the data source

To create a data source, which is used for event tracking and recovery during ALE inbound processing, you use the administrative console. You select a JDBC provider and then create a data source in the JDBC provider. After configuring the data source, you use the Test Connection button in the administrative console to test the database connection.

Before you begin

Before configuring the data source, make sure the database is already created and then configure the data source using that database.

About this task

You need a JDBC provider only if you are going to set up an event recovery table to persist inbound events (to ensure once-only delivery).

Procedure

1. In the administrative console, select a JDBC provider.
   a. Click **Resources** → **JDBC** → **JDBC Providers**.
   b. Select a JDBC provider.
2. Select **Data sources**.
3. Create a new data source by clicking **New**.
4. Type values for the required fields.
   a. In the **Data source name** field, type the name of the event table.

      A default value is provided. For example, for the Derby JDBC Provider, the default value is **Derby JDBC Driver DataSource**. You can change the default value.

      An example of a data source name is: `EventRecoveryDS`
   b. In the **JNDI Name** field, type the JNDI name of the data source.

      An example is `jdbc/EventRecovery`.
5. Optionally, select an authentication alias for the JDBC provider from the **Component-managed authentication alias and XA recovery authentication alias** list.
6. Click **Next**.
7. In the Create a data source window, indicate the database to which the data source connects by typing a value in the **Database name** field.
8. Review the information in the Summary table to ensure its accuracy, and then click **Finish**.

9. Save your configurations.
10. From the list of data sources, select the check box next to the data source that you created in the previous steps.
11. Click **Test connection**.

    You will see a message that the test was successful.

    Note: If the test is not successful, make sure the database drivers are available in lib\ext directory. Also make sure that the database name and port are correct.

Results

A new data source is created.

What to do next

Configure the adapter for ALE inbound processing. Use the database JNDI created in this topic, and use the Auto create event table property to create the event recovery table.

Creating an IDoc definition file

When you configure the adapter for ALE processing, you typically have the J2C Bean wizard create a business-object definition based on the IDoc or IDocs it finds on the SAP system. Alternatively, you can have the J2C Bean wizard generate the business-object definition based on an IDoc definition file, which you create.

About this task

Use the following general procedure to create the IDoc definition file. Note that the steps for generating these definitions can vary from system release to release. For example, on some versions of the SAP server, you might need to clear the **IDoc record types** check box if it is checked.

Note: Follow this procedure only if you plan to use the **Discover IDoc from File** choice in the J2C Bean wizard. If you plan to use **Discover IDoc from System**, you do not need to create an IDoc definition file.

Procedure

1. In the SAP user interface, select transaction WE63 by entering `/oWE63`.
2. In the **Basic type** field, enter the basic IDoc type (for example, `ALEREQ01`) or browse to see a list of basic types.
3. Click **Documentation → Parser** or click the Parser icon.

   The IDoc definition is displayed on the screen.
4. Save the definition to a directory on your local file system by clicking **System → List → Save → Local File**.
5. From the Save list in file window, select **unconverted** and select the check icon.

   Note that **unconverted** is the only supported format.
6. Enter the location where the file should be saved (or browse to the location) and click **Generate**.

Results

The IDoc definition file is located on your local file system.

What to do next

Configure the adapter for ALE outbound or inbound processing.

Adding transport files to the SAP server

To use the Advanced event processing interface, you must first add the adapter-supplied transport files to the SAP server.

About this task

Note: This procedure is for the Advanced event processing interface only. If you are not using the Advanced event processing interface, skip this procedure.

The transport files for WebSphere Adapter for SAP Software contain a variety of objects, such as table structures, functions, and data. These development objects must be imported into the SAP server before you can use the advanced event processing interface.

The transport files are provided as .zip files in the Rational Application Developer for WebSphere Software installation directory. The path to the files within that directory is ResourceAdapters\SAP_6.1.0.0_*xx*>\transports.

From within transports, the files are located in one of the following directories:

- transports_40_45_46, which you use with SAP version 4.0, 4.5, or 4.6
- transports_47 _erp, which you use with SAP version 4.7 and above

Procedure

1. Create the namespace for the adapter before installing the transport files. Provide the following name for the namespace: `/CWLD/`
2. Import the transport files into the SAP server in the order shown:
   a. CWYAP_SAPAdapter_AEPTransport_Infrastructure.zip
   b. CWYAP_SAPAdapter_AEPTransport_Primary.zip

Results

The files needed to use Advanced event processing are installed on the SAP server.

What to do next

Configure the adapter for Advanced event processing.

Implementing event-detection mechanisms

When you use the Advanced event processing interface for inbound processing, you must determine an event-detection mechanism for the business process with which you are working. You then implement that process.

About this task

Note: These procedures are for the Advanced event processing interface only. If you are not using the Advanced event processing interface, skip the procedures.

Sample code and examples are provided to help you implement an event-detection mechanism.

Implementing custom triggers

Custom triggers requires encapsulating a portion of ABAP code in a custom function module. The event-detection code is written as a function module to ensure that the processing remains separate from the transaction. Any tables or variables used from the transaction must be passed to the function module by value and not by reference.

About this task

Note: This procedure is for the Advanced event processing interface only. If you are not using the Advanced event processing interface, skip this procedure.

To minimize the effects of locking a business object when retrieving an event, the function module typically executes in an update-task mode. To avoid inconsistencies, do not use update task if the function module is already being called within a process that is in an update-task mode.

To minimize the impact in the transaction, place the function module within another include program. Using an include program allows you to make changes to custom code rather than to SAP code.

The event-detection code contains logic that identifies the object for the event. For example, the sales order transaction handles many types of orders, but only one order type is required. This logic is in the event-detection code. The general strategy for placing this event-detection code is to insert it just before the data is committed to the database. The function module containing the event detection code is typically created as a part of the function group for the business object.

To implement a custom trigger for event detection:

Procedure

1. Determine which verbs to support: Create, Update, or Delete. This helps define which transactions to investigate.
2. Determine the business-object key for the transaction. This key must be unique to allow the adapter to retrieve the business object from the database.

   If a composite key is required, at triggering time you can specify each key attribute and its corresponding value as a name-value pair. When the business object is created at polling time, the adapter automatically populates the attributes with their values.
3. Check that an SAP-provided user exit in the transaction has all the information needed to detect an event.

   For example, a user exit might not be able to implement a Delete verb because the business object is removed from the database before that point.
4. If a user exit cannot be used, determine the appropriate location for the event-detection code, and then add the event-detection code using an SAP modification. Select a location that has access to the business object key and other variables used to make the decision. If you are implementing the future events capability, in addition to adding the event-detection code for future events, contact your Basis administrator to schedule the adapter-delivered batch program /CWLD/SUBMIT_FUTURE_EVENTS to run once every day.
5. Research a business process by looking for a “commit work statement” in the code executed by the transaction for the business process. You can use the ABAP debugger to investigate the value of different attributes at that point.
6. Determine the criteria for detecting an event.

7. Create the function module containing the event detection code.
8. Create the include program and then add it to the transaction's code.
9. Test all of the scenarios designed to detect an event.

Example

The following steps describe the process of creating an example SAP customer master using the custom trigger event-detection mechanism. The code that follows it is a result of this process.

1. Upon investigation of the SAP customer master transaction, transaction XD01 is found to support the desired customer master creation business process.
2. The Customer number is determined to be the unique key. The Customer number is stored in table/field KNA1-KUNNR.

   Note: Because this event uses a single unique key, the code example uses the OBJKEY parameter to pass the key value.
3. Transaction XD01 has a user exit in the transaction flow as part of the document save process (Form Userexit_save_document). At this point in the transaction, the customer number is available when the user exit is executed.
4. An include statement is added to the user exit that points to the include program.
5. At this time, the include program and a function module must be created.

The following code fragment illustrates the function call to the /CWLD/ADD_TO_QUEUE_AEP event trigger (using a single key value).

```
CASE HEADER_CHANGE_IND.
 WHEN 'I'.
 * The verb will always be a create if KNA1 data is entered.
  IF KNA1_CREATE = 'X'.
  HEADER_EVENT = C_CREATE_EVENT.
 ELSE.
 * Check if an entry is in the config table for converting a create. If
 * no entry is found, the default is to convert the extension of sales
 * area or company code to an update.
 SELECT SINGLE * FROM /CWLD/CONF_VAL
  WHERE CONF_NAME = C_CONVERT_CREATE
  AND CONF_VALUE = C_FALSE_WORD.

 IF SY-SUBRC = 0.
  HEADER_EVENT = C_CREATE_EVENT.
 ELSE.
  HEADER_EVENT = C_UPDATE_EVENT.
 ENDIF.
 ENDIF.

 WHEN 'U'.
  HEADER_EVENT = C_UPDATE_EVENT.
 WHEN 'E' OR 'D'.
  HEADER_EVENT = C_DELETE_EVENT.
ENDCASE.

 * See if it's a sold-to company.
 SELECT SINGLE * FROM /CWLD/CONF_VAL
  WHERE CONF_NAME = C_AGCUSTMASTER
  AND CONF_VALUE = KNA1-KTOKD.

 * clear temp_obj_type.
 CLEAR TEMP_OBJ_NAME.
 IF SY-SUBRC = 0.
 * temp_obj_type = 'YXR_V51'.
```

```
  TEMP_OBJ_NAME = C_OBJ_CUSTOMERMASTER.
 ELSE.

* If it's not a sold-to company, check if it's another partner.
 SELECT SINGLE * FROM /CWLD/CONF_VAL
  WHERE CONF_NAME = C_AGCUSTPARTNER
  AND CONF_VALUE = KNA1-KTOKD.
 ENDIF.

CALL FUNCTION  '/CWLD/ADD_TO_QUEUE_AEP'
 EXPORTING
  OBJ_NAME = TEMP_OBJ_NAME
  OBJKEY = OBJKEY
  EVENT = HEADER_EVENT
* IDOC_NUMBER =
  GENERIC_RECTYPE = GENERIC_RECTYPE
 IMPORTING
  RECTYPE = RECTYPE
 TABLES
  EVENT_CONTAINER  = EVENT_CONTAINER
 EXCEPTIONS
  OTHERS = 1.
```

The following code fragment illustrates the function call to the /CWLD/ADD_TO_QUEUE_IN_FUT_AEP event trigger (single key value).

```
DATA: DATE_IN_FUTURE LIKE SY_DATUM.

CALL FUNCTION ' /CWLD/ADD_TO_QUEUE_IN_FUT_AEP'
 EXPORTING
  OBJ_NAME = TEMP_OBJ_NAME
  OBJKEY = OBJKEY
  EVENT = HEADER_EVENT
  VALID_DATE = DATE_IN_FUTURE
 IMPORTING
  RECTYPE = RECTYPE
 TABLES
  EVENT_CONTAINER = EVENT_CONTAINER
 EXCEPTIONS
  OTHERS = 1.
```

What to do next

Configure the adapter for Advanced event processing.

Implementing batch programs

To implement batch program as an event detection mechanism, you must write an ABAP program that evaluates database information. If the criteria in the ABAP program is fulfilled when the program executes, then an event is triggered.

About this task

Note: This procedure is for the Advanced event processing interface only. If you are not using the Advanced event processing interface, skip this procedure.

To implement batch program for event detection:

Procedure

1. Determine which verb to support: Create, Update, or Delete.
2. Determine the business object key for the transaction.

   The business object key must be unique so that the business object can be retrieved from the database. A composite key might be required.

3. Determine the criteria for detecting an event.

   You should have knowledge of the database tables associated with a business object.
4. Create an ABAP program containing the criteria for generating an event.
5. If you are implementing the future events capability, in addition to adding the event-detection code for future events, contact your Basis administrator to schedule the adapter-delivered batch program /CWLD/ SUBMIT_FUTURE_EVENTS to run once every day.
6. Determine if a background job is required to automate the batch program.

   A background job is useful if there is an impact on system resources, which makes it necessary to run the batch program during off-peak hours.

Example

The following steps describe the process of creating a batch program that detects events for all sales quotes created on today's date. The code that follows it is a result of this process.

1. Create is determined to be the supported verb.
2. The quote number is determined to be the unique key used to retrieve the events.
3. The creation date (VBAK-ERDAT) and the document category (VBAK-VBTYP) must be checked.

The following sample code supports the SAP sales quote as a batch program:

```
REPORT ZSALESORDERBATCH.
tables: vbak.

parameter: d_date like sy-datum default sy-datum.

data: tmp_key like /CWLD/LOG_HEADER-OBJ_KEY,
   tmp_event_container like swcont occurs 0.

" retrieve all sales quotes for today's date

" sales quotes have vbtyp = B

select * from vbak where erdat = d_date and vbtyp = 'B'.

tmp_key = vbak-vbeln.

CALL FUNCTION '/CWLD/ADD_TO_QUEUE_AEP'
 EXPORTING
  OBJ_NAME = 'SAP4_SalesQuote'
  OBJKEY = tmp_key
  EVENT = 'Create'
  GENERIC_RECTYPE = ''
 IMPORTING
  RECTYPE = r_rectype
 TABLES
  EVENT_CONTAINER = tmp_event_container.

write: / vbak-vbeln.

endselect.
```

What to do next

Configure the adapter for Advanced event processing.

Implementing business workflows

Business workflow is a set or sequence of logically related business operations. The processing logic within a workflow detects events. The business workflow event-detection mechanism relies on the SAP Business Object Repository (BOR), which contains the directory of objects along with their related attributes, methods, and events.

About this task

Note: This procedure is for the Advanced event processing interface only. If you are not using the Advanced event processing interface, skip this procedure.

To implement business workflow for event detection:

Procedure

1. Determine which SAP business object represents the functionality that you need. Check if the events trigger, start, or end a workflow.

   You can use the Business Object Builder (transaction SWO1) to search for the appropriate business object.
2. Create a subtype of this SAP business object.

   A subtype inherits the properties of the supertype and can be customized for use.
3. Activate the events (such as CREATED, CHANGED, and DELETED) for the business object by customizing the subtype.

Example

The following example of SAP sales quote can be used to implement an event trigger using business workflow:

1. Search the BOR for the appropriate sales quote business object. A search can be done using the short description field and the string '*quot*'. BUS2031 (Customer Quotes) is one of the business objects returned.
2. Upon further investigation of BUS2031, it is determined that the key field is CustomerQuotation.SalesDocument (VBAK-VBELN).
3. A subtype for BUS2031 is created using the following entries:
   - Object type—ZMYQUOTE
   - Event—SAP4_SalesQuote
   - Name—SAP4 Sales Quote
   - Description—Example of an SAP 4 Sales Quote Subtype
   - Program—ZMYSALESQUOTE
   - Application—V
4. The event detection mechanism is activated by adding an entry to the Event Linkage table (transaction SWE3). The create event is activated using the following entries:
   - Object type—ZMYQUOTE
   - Event—SAP4_SalesQuote
   - Receiver FM— /CWLD/ADD_TO_QUEUE_DUMMY_AEP
   - Receiver type FM— /CWLD/ADD_TO_QUEUE_WF_AEP

Note: The Receiver and Receiver type function modules (FM) point to /CWLD/ADD_TO_QUEUE_AEP. The DUMMY function module is used only

because sometimes the SAP application requires that both fields be populated. The WF function module translates the SAP standard interface to the one used by /CWLD/ADD_TO_QUEUE_AEP.

The business workflow event-detection mechanism is created and active. It is set up to detect all SAP Customer Quotes that are created.

What to do next

Configure the adapter for Advanced event processing.

Implementing change pointers

A change pointer uses change documents and is one of the more challenging event detection mechanisms to implement. The SAP Business Object Repository (BOR) is used as well as Application Link Enabled (ALE) technology. A change document always refers to a business document object having at least one database table assigned to it. If the data element in a table is marked as requiring a change document and the table is assigned to a business document object, then a change in value of the field defined by the data element generates a change document. The changes are captured in tables CDHDR and CDPOS and are used for event detection.

About this task

Note: This procedure is for the Advanced event processing interface only. If you are not using the Advanced event processing interface, skip this procedure.

To implement change pointer for event detection:

Procedure

1. Activate the global Change pointers flag in transaction BD61.
2. Change the SAP function module CHANGE_POINTERS_CREATE to include the function module call to /CWLD/EVENT_FROM_CHANGE_POINTR.
3. Determine which verbs to support: Create, Update, or Delete.
4. Check if the SAP business process (transaction) utilizes change documents:
   - In the Environment menu for the transaction, does a Change function exist? How about when you click Go To, and then click Statistics?
   - If you change data in the transaction, is there a new entry in table CDHDR that reflects the change?
   - In the database tables associated with a transaction, do any of the data elements have the Change Document flag set?
5. If the answer is Yes to any of these questions, the transaction uses change documents.
   a. Determine if the data elements that set the Change Document flag capture all of the information needed to detect an event. Changing the Change Document flag is not recommended because it changes an SAP-delivered object.
   b. Determine the business object key for the transaction. The business object key must be unique so that the business object can be retrieved from the database. A composite key may be required. This is normally table/field CDHDR-OBJECTID.

c. Determine the criteria for detecting an event. Use table/field CDHDR-OBJECTCLAS as the main differentiator. CDPOS-TABNAME may also be used to detect the event.

d. Update function module /CWLD/EVENT_FROM_CHANGE_POINTR with the logic to detect the event.

Example

The following example of an SAP sales quote can be used to implement an event trigger using change pointer:

1. Update is determined to be the supported verb. Investigating the sales quote create transaction shows that the Create verb is not detected through this mechanism.
2. When performing the checks of the business for sales quote:
   - The Change function is available from the Environment menu in transaction VA22.
   - Making a change to a sales quote results in a new entry in table CDHDR.
   - Looking at table VBAP, the field ZMENG has the Change Document flag set.
3. No evaluation of data elements was done for this example.
4. The sales quote number is determined to be the unique key in CDHDR-OBJECTID.
5. CDHDR-OBJECTCLAS has a value of VERKBELEG, which is the main differentiator. Only sales quotes should be picked up. The code checks the TCODE field in the header table, but a proper lookup should be done in the VBAK table.

The following sample code is added to /CWLD/ EVENT_FROM_CHANGE_POINTR:

```
when 'VERKBELEG'.
 data: skey like /cwld/log_header-obj_key,
 s_event like swetypecou-event,
 r_genrectype like swetypecou-rectype,
 r_rectype like swetypecou-rectype,
 t_event_container like swcont occurs 1 with header line.

" Quick check. Should check document category (VBTYP) in VBAK.
check header-tcode = 'VA22'.

" Event detection has started
perform log_create using c_log_normal c_blank c_event_from_change_pointer c_blank.

" Set the primary key
skey = header-objectid.

" Set the verb
s_event = c_update_event.

" Log adding the event to the queue
perform log_update using c_information_log text-i44
 'SAP4_SalesQuote' s_event skey.

" Event detection has finished.
perform log_update using c_finished_log c_blank
 c_blank c_blank c_blank.

call function '/CWLD/ADD_TO_QUEUE_AEP'
 exporting
  obj_name = 'SAP4_SalesQuote'
  objkey = skey
```

```
  event = s_event
  generic_rectype = r_genrectype
 importing
  rectype = r_rectype
 tables
  event_container = t_event_container
 exceptions
  others = 1.
```

What to do next

Configure the adapter for Advanced event processing.

Launching the J2C Bean wizard

To begin the process of creating and deploying a module, you start the J2C Bean wizard in Rational Application Developer for WebSphere Software. The wizard creates a connector project, which is used to organize the files associated with the module.

Before you begin

Make sure you have gathered the information you need to establish a connection to the SAP server. For example, you need the name or IP address of the SAP server and the user ID and password needed to access it.

About this task

If you have an existing project, you can use it instead of creating a new one. Select it before you start the wizard.

Procedure

1. To start the J2C Bean wizard, go to the Enterprise Explorer of Rational Application Developer for WebSphere Software and click **File** → **New** → **Other** → **J2C** → **J2C Bean**.
2. Click **Next**.
3. In the Resource Adapter Selection window, expand the SAP folder and select **IBM WebSphere Adapter for SAP Software (IBM : *version*)**, where *version* is the version of the adapter you want to use, for example, 6.2.0.0
4. Click **Next**.
5. In the window, accept the default project name in the **Connector project** field or type a different name.
6. In the **Target server** field, select the type of server where you will deploy the module. The wizard creates the artifacts that are appropriate to that server.
7. Click **Next**. The Connector Settings window is displayed.

Results

A new connector project is created, which contains the adapter RAR file. The project is listed in the Enterprise Explorer view of the Java EE perspective.

What to do next

Continue working in the J2C Bean wizard. The next step is to add database-specific files to the project.

Configuring connector dependencies

As part of generating the service, you are prompted by the J2C Bean wizard to specify the location of the required sapjco.jar file and related files.

About this task

To obtain the required files and specify their location, use the following procedure.

Procedure

1. Obtain the sapjco.jar file and the associated files for your operating system from your SAP administrator or from the SAP Web site. The files are listed in the following table.

Table 10. External software dependency files required by SAP Software

Operating system	Files to be copied
Windows® and i5/OS®	sapjco.jar and any *.dll files that come with the SAP JCo download from the SAP Web site
UNIX® (including UNIX System Services on z/OS®)	sapjco.jar and any .so and .o files that come with the SAP JCo download from the SAP Web site

2. SAP JCo requires msvcp71.dll and msvcr71.dll on Windows environment. These dlls are found in the system32 directory on most Windows systems. Copy these dlls onto your Windows environment if it does not have them.
3. From the Required Files and Libraries window, specify the location of the files:
   a. For each file, click **Browse** and select the location of the file.
   b. Click **Next**.

Results

The sapjco.jar file and associated files are now part of your project.

Configure the adapter. The first step in the process of configuring the adapter is to specify information about the SAP server so that the J2C Bean wizard can establish a connection to the server.

Setting connection properties for the J2C Bean wizard

To set connection properties for the J2C Bean wizard so that it can access the SAP server, specify such information as the user name and password you use to access the server as well as the name or IP address of the server.

Before you begin

Make sure you have successfully added the external dependency files (the sapjco.jar and associated files).

About this task

Specify the connection properties that the J2C Bean wizard needs to establish a connection to the SAP server and discover functions or data.

To specify the connection properties, use the following procedure.

Procedure

1. From the Processing Direction window, perform the following steps:
   a. Select **Inbound** (if you are going to send data from the SAP server) or **Outbound** (if you are going to send data to the SAP server).
   b. Click **Next**.
2. From the Discovery Configuration window, specify the configuration properties:
   a. In the **Host name** field, type the name (or IP address) of your SAP server.
   b. Optionally, change the default value for **System number**.
   c. Type your client ID (or use the default value if your client ID is 100).
   d. If necessary, change the default setting for **Language code** by clicking **Select** and selecting a value from the list.

      The default value in the **Code page** field is related to the value in the **Language code** field. For example, if the language code is `EN (English)`, the code page number is `1100`. If you change the language code to `TH (Thai)`, the code page number changes to `8600`.
   e. Type the name and password you use to access the SAP server.

      The password is case-sensitive.
   f. Select an interface from the **SAP interface name** list.
3. To set additional advanced properties, click **Advanced**. The related properties are displayed.
4. To set RFC tracing properties, perform the following steps:
   a. Expand **SAP RFC trace configuration** and select **RFC trace on**.
   b. Select a tracing level from the **RFC trace level** list.
   c. Click **Browse** and select a location to which the RFC trace files will be saved.
5. If you need to set bidirectional properties, perform the following steps:
   a. Expand **Bidi properties** and select the **Bidi transformation** check box.
   b. Set the properties, ordering schema, text direction, symmetric swapping, character shaping, and numeric shaping to control how bidirectional transformation is performed.
6. To change the location of the wizard's log files or the amount of information included in the logs, click the **Specify the level of the logging desired** check box. Provide the following information:
   a. In the **Log file output location** field, specify the location of the log file for the wizard. Click **Browse** if you want to select a different location.
   b. In the **Logging Level** field, specify the severity of errors that you want logged. This log information is for the wizard only; at run time, the adapter writes the error messages and the trace information into the standard log and trace files for the server.

      Note: This log pertains to the J2C Bean wizard only, not to the operation of the adapter.
7. Click **Next**.

Results

The J2C Bean wizard contacts the SAP server, using the information you provided (such as user name and password) to log in. You see the Object Discovery and Selection window.

Specify search criteria that the J2C Bean wizard uses to discover functions or data on the SAP server.

Configuring the module for outbound processing

To configure a module to use the adapter for outbound processing, use the J2C Bean wizard in Rational Application Developer for WebSphere Software to find and select business objects and services from the SAP server, and to generate business object definitions and related artifacts.

Configuring the J2C Bean and Java data bindings for the BAPI interface

To configure a module to use the adapter for BAPI outbound processing, you use the J2C Bean wizard in Rational Application Developer for WebSphere Software to find a BAPI or set of BAPIs. You then configure the business objects that are generated and create a deployable module.

Selecting objects for the BAPI interface

To specify which BAPI function or functions you want to call and which data you want to process, you provide information in the J2C Bean wizard.

Before you begin

Make sure you have set the connection properties for the J2C Bean wizard.

About this task

Specify search criteria that the J2C Bean wizard uses to discover BAPI functions on the SAP server. The J2C Bean wizard returns a list of BAPI functions that meet the search criteria.

To specify the search criteria and select one or more BAPI functions, use the following procedure.

Procedure

1. In the Object Discovery and Selection window, indicate which BAPI or set of BAPIs you want to work with.
   a. Click **RFC** to enable the filter button.
   b. Click the filter button.

   Note: Instead of using the filter capability, you can expand **RFC** and select the function from the list, or you can expand **BOR**, expand the functional grouping (for example, **Cross-Application Components**), and select the BAPI. Then skip ahead to step 4 on page 71.
2. From the Filter Properties window, specify information about the BAPI or BAPIs you want to discover:
   a. Select **Discover objects by name** or **Discover objects by description** from the **Object attribute to use for discovery** list.
   b. Type a search string (for example, BAPI_CUSTOMER*) representing the BAPI you want to call.

   This is the name of the BAPI in SAP plus an asterisk as a wild card character to indicate that you want a list of all SAP application components that start with the phrase BAPI_CUSTOMER.

c. Indicate the number of functions you want returned by changing the value in the **Maximum number of objects to retrieve** field or by accepting the default value.
d. Click **OK**.

3. Select the BAPI or BAPIs.
   a. Expand **RFC (filtered)**.
   b. Click the BAPI you want to use. If you are working with multiple BAPIs, click the names of all the BAPIs.
4. Click the arrow button to add the BAPI or BAPIs to the **Selected objects** list.
5. In the Configuration Properties window, perform the following steps for each BAPI to add it to the list of business objects to be imported:
   a. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   b. If the BAPI has optional parameters associated with it, expand **Optional parameters**, and select the type of parameters (import, export, or table) that you want to work with.

      By default, the J2C Bean wizard generates all the parameters required for the selected BAPI, so select this check box and then clear the check boxes for any parameters you do not want to include in your business object.

      For example, if you are adding the ChangeFromData BAPI, you have the option of adding the following parameters:

      PI_DIVISION

      PI_DISTR_CHAN

      Refer to the SAP documentation for a list and description of the optional parameters.
   c. Click **OK** to add the BAPI to the list of business objects to be imported.

   If you want to remove an object from the list, select the object name and click the left arrow.
6. Click **Next**.

Results

The J2C Bean wizard has returned the function or functions that match the search criteria, and you have selected the function or functions you want to work with. The Configure Composite Properties window is displayed.

What to do next

Provide information about the business object (such as the name of the top-level object and associated operation).

Configuring the Java data bindings for the BAPI interface

To configure simple BAPI business objects, you specify information about the object (such as the name of the object and the operation associated with the object). If you are using the version of the adapter with transaction support, you also select the type of remote function call you want to make (**Synchronous RFC**, **Asynchronous transactional RFC**, or **Asynchronous queued RFC**).

Before you begin

If you want to use the **Asynchronous Transactional RFC** or **Asynchronous Queued RFC** choice, you must have installed IBM WebSphere Adapter for SAP Software with transaction support (CWYAP_SAPAdapter_Tx).

If you are sending the function call to a queue on the SAP server (so that an application on SAP server can process the BAPIs SAP server in order), make sure you have configured the queue on the SAP server.

About this task

To configure the business object, use the following procedure.

Procedure

1. In the Configure Composite Properties window, select a name for the top-level business object.
2. Perform one of the following sets of tasks:
   - If you are working with a single BAPI, click **Add**, select an operation (for example, **Retrieve**), and click **OK**.

     You can select only one operation for the BAPI.
   - If you are working with multiple BAPIs, select, for each operation, the BAPI you want associated with that operation, as described in the following steps:
     a. Click **Add**, select the operation (for example, **Create**) from the list, and click **OK**.
     b. From the **RFC function for selected operation** list, select a BAPI to associate with the operation you selected in the previous step.
     c. For the second BAPI, click **Add**, select an operation (for example, **Retrieve**) from the list, and click **OK**.
     d. From the **RFC function for selected operation** list, select a BAPI to associate with the operation you selected in the previous step.
     e. For any subsequent BAPIs, repeat the previous two steps.

     You can select only one operation per BAPI.
3. In the **Business object namespace** field, use the default namespace (http://www.ibm.com/xmlns/prod/websphere/j2ca/sap) except in the following circumstance. If you are adding the business object to an existing module and the module already includes that business object (from an earlier run of the J2C Bean wizard), change the namespace value.

   For example, you could change the namespace to http://www.ibm.com/xmlns/prod/websphere/j2ca/sap1.
4. To indicate where the business object information should be stored, type the path to the location in the **Folder** field. This is an optional step.
5. If you are using the version of the adapter with transaction support, you can select the type of remote function call you want to make.

   Note: If you are using the version of the adapter without transaction support (CWYAP_SAPAdapter), this step does not apply. The BAPI or BAPIs are sent synchronously. Skip ahead to step 6 on page 73

   If you are using the version of the adapter with transaction support (CWYAP_SAPAdapter_Tx) but do not select a type of remote function call, the default (**Synchronous RFC**) is used. In Synchronous RFC, the adapter calls the BAPI and then waits for the response from the SAP server.

   a. Select the arrow next to the **SAP Remote Function Call (RFC) type** list.

b. Select one of the RFC types:
   - Select **Synchronous RFC** (the default) if you want the BAPI sent in a synchronous manner (the adapter calls the BAPI and then waits for the response from the SAP server). Note that the receiving system must be available when you use **Synchronous RFC**.
   - Select **Asynchronous Transactional RFC** if you want the call to succeed regardless of whether the receiving system (the SAP server) is available.
     – If the event is successful, the adapter sends the transaction ID to the client.
     – If the event fails, the adapter returns an AbapException with the transaction ID to the adapter client. The adapter client can use this transaction ID to make the call again at a later time.

     Note: When you use **Asynchronous Transactional RFC**, no data is returned to the client from the adapter.
   - Select **Asynchronous Queued RFC** if you want the BAPI or BAPIs delivered to a predefined queue on the SAP server. After you select**Asynchronous Queued RFC**, select, from the list, the specific queue on the SAP server to which the BAPI or BAPIs will be delivered.

     If no queues exist on the SAP server, you can type the name of a queue. You can then create the queue on the SAP server after configuration.

     Note: If you do not select a queue, the adapter will configure the object to use the **Asynchronous Transactional RFC** type.

6. If you want to continue to process a BAPI even if the BAPI return object contains errors, select the **Ignore errors in BAPI Return object** check box.

   Note: If you selected **Asynchronous Transactional RFC** or **Asynchronous Queued RFC**, this check box is not available.
7. Click **Finish**.

Results

You specified a name for the top-level business object, selected an operation for the BAPI or BAPIs, and indicated the type of remote function call. The Service Generation and Deployment Configuration window is displayed.

What to do next

Generate a deployable module that includes the adapter and the business objects.

Setting deployment properties and generating artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **Project name** field, select or create a new project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the New Source Project Creation window, select **Java project**.
     c. In the Create a new project window, type a name for the project. For example, MyApdapterOutbound.
     d. Accept the default values for the other fields.
     e. Click **Finish**.
2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new package:
     a. Click **New**.
     b. In the Java Package window, click **Java project**.
     c. In the Create a new project window, type a name for the package. For example, myadapteroutboundpkg.
     d. Accept the default values for the other fields.
     e. Click **Finish**.
3. In the **Interface name** field, specify the interface name you want to use for your business objects. For example, MyAdapterOutboundInterface. The implementation name is automatically generated by suffixing ″Impl″ to the interface name and the name is displayed in the **Implementation name** field. For example, MyAdapterOutboundInterfaceImpl.
4. Optional: Select the **Enable generate Command Bean** check box, to generate a command bean for every operation that you have selected. If you create command bean, you need to specify the command bean name as well as the input and output names.
5. In the Connection properties area, specify how you want the adapter to connect to the database.
   - To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.
   - To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.
6. If you select **Managed Connection (recommended)** check box, specify how you want the adapter to specify the connection properties.
   - To select an existing name, click **Browse**.
   - Otherwise, create a new name.
     a. Click **New**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field. For example, `com/eis/AdapterCF`.
     d. Click **Finish**.

7. If you select the **Non-managed Connection** check box, the Database system connection information area is expanded to show the connection information. Review the connection information and change the values if required. You can change the values to specify a different user name and password at run time. You can also connect to an alternate database, although the schema names must be the same in both databases. The format of the connection properties is database-specific.
8. Optional: Specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties.
9. When you are finished setting properties, click **Finish**.

Results

The new project is added to the Enterprise Explorer perspective. The module is created in the project and artifacts are generated.

The generated artifacts allow you to build an enterprise application that accesses the EIS. You can use the J2C Bean and Java data bindings directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to the Rational Application Developer documentation.

Generating the EJB or JSP project

After you create the Java project, use the Web Page, Web Service, or EJB from J2C Java wizard to create the EJB or JSP project.

About this task

Procedure

1. To start the J2C Bean wizard, go to the Enterprise Explorer of Rational Application Developer for WebSphere Software and click **New>Other>J2C> Web Page, Web Service, or EJB from J2C Java Bean**.
2. Click **Next**.
3. In the J2C Java bean selection window, click **Browse**.
4. In the Find J2C Bean window, type the first letter of the implementation name generated earlier or type the full name and press Enter.
5. Select the implementation name from the list and click **OK**.
6. Click **Next**.
7. In the Deployment Information window, select the Java EE Resource Type as **EJB** or **Sample JSP** and click **Next**.

   Note: In the Deployment Information window, the **Check Configure Resource Adapter Deployment** check box is available for selection only if you have selected the **Non-managed Connection** check box when specifying the deployment settings.
8. If you select **EJB**, the EJB Creation wizard is displayed. This wizard creates the Java project as an EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.

b. In the EJB Project window, type an EJB project name. For example, MyAdapterOutboundEJB.
c. In the EAR Membership area, click **New** to create a new ear project name.
d. In the EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundEJBEAR.
e. Click **Next**.
f. In the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
g. Select the **Generate Deployment Descriptor** check box.
h. Click **Finish** to return to the EJB Project window.
i. Click **Finish** to return to the EJB Creation window.
j. Click **Next**.

Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the EJB and EAR project and return to EJB Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.

k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the EJB EAR Project you created.
l. Click **Finish**.

9. If you select **Sample JSP**, the Simple JSP Creation wizard is displayed. .
   - To select an existing project, click **Browse**. If the desired project name appears in the Project Selection list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In Dynamic Web Project window, type a project name. For example, MyAdapterOutboundWEB.
     c. In EAR Membership area, click **New** to create a new ear project name.
     d. In EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundWEBEAR.
     e. Click **Next**.
     f. In the Java EE Module dependencies area of the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the Dynamic Web Project window.
     i. Click **Finish** to return to the Simple JSP Creation window.
     j. Click **Next**.

     Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the WEB and EAR project and return to Simple JSP Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.

     k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the Web EAR Project you created.

l. Click **Finish**.

10. Export the project as an EAR file for deployment.

Configuring a module for the BAPI work unit interface

To configure a module to use the adapter for BAPI work unit processing, you use the J2C Bean wizard in Rational Application Developer for WebSphere Software to find a set of BAPIs. You then configure the business objects that are generated and create a deployable module.

Selecting business objects and services for BAPI work unit processing

To specify which BAPI functions you want to call and which data you want to process, you provide information in the J2C Bean wizard.

Before you begin

Make sure you have set the connection properties for the J2C Bean wizard.

About this task

Specify search criteria that the J2C Bean wizard uses to discover BAPI functions on the SAP server. The J2C Bean wizard returns a list of BAPI functions that meet the search criteria.

To specify the search criteria and select the BAPI functions for the work unit, use the following procedure.

Procedure

1. In the Object Discovery and Selection window, indicate which BAPI you want to work with.
   a. Click **RFC** to enable the filter button.
   b. Click the filter button.

      Note: Instead of using the filter capability, you can expand **RFC** and select the function from the list, or you can expand **BOR**, expand the functional grouping (for example, **Cross-Application Components**), and select the BAPI. Then skip ahead to step 4 on page 78.
2. From the Filter Properties window, specify information about the BAPIs you want to discover:
   a. Select **Discover objects by name** or **Discover objects by description** from the **Object attribute to use for discovery** list.
   b. Type a search string (for example, BAPI_CUSTOMER*) representing the BAPI you want to call.

      This is the name of the BAPI in SAP plus an asterisk as a wild card character to indicate that you want a list of all SAP application components that start with the phrase BAPI_CUSTOMER.
   c. Indicate the number of functions you want returned by changing the value in the **Maximum number of objects to retrieve** field or by accepting the default value.
   d. Click **OK**.
3. Select the BAPIs.
   a. Expand **RFC (filtered)**.

b. Click the BAPIs you want to make part of the work unit.

4. Click the arrow button to add the BAPIs to the **Selected objects** list.
5. In the Configuration Properties window, perform the following steps for each BAPI to add it to the list of business objects to be imported:
   a. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   b. If the BAPI has optional parameters associated with it, select the **Select optional parameters to include as child objects** check box, expand **Optional parameters**, and select the type of parameters (import, export, or table) that you want to work with.

      By default, the J2C Bean wizard generates all the parameters required for the selected BAPI, so select this check box and then clear the check boxes for any parameters you do not want to include in your business object.

      For example, if you are adding the ChangeFromData BAPI, you have the option of adding the following parameters:

      PI_DIVISION

      PI_DISTR_CHAN

      Refer to the SAP documentation for a list and description of the optional parameters.
   c. Click **OK** to add the BAPI to the list of business objects to be imported.

   If you want to remove an object from the list, select the object name and click the left arrow.
6. Click **Next**.

Results

The J2C Bean wizard has returned the functions that match the search criteria, and you have selected the functions you want to work with. The Configure Composite Properties window is displayed.

What to do next

Provide information about the business objects (such as the name of the top-level object and associated operation).

Configuring BAPI work unit objects

To configure a BAPI work unit business object, you specify information about the object (such as the name of the object, the operations associated with the BAPIs in the work unit, and the sequence in which you want the BAPIs to be processed).

Before you begin

Make sure you have selected and imported the BAPI functions.

About this task

To configure the business object, use the following procedure.

Procedure

1. In the Configure Composite Properties window, select a name for the top-level business object.

2. Associate an operation with each BAPI and specify the order in which the BAPIs should be processed:
   a. Click **Add**, select an operation (for example, **Create**), and click **OK**.
   b. In the **Sequence of RFC functions for the selected operation** section of the window, indicate the order in which the BAPIs should be processed by clicking **Add**, selecting the BAPI that should be processed first, and clicking **OK**.
   c. For each subsequent BAPI in the transaction, click **Add**, select the BAPI, and click **OK**.
   d. After you have added all the BAPIs, click **Add**, select **COMMIT**, and click **OK**.
3. In the **Business object namespace** field, use the default namespace (http://www.ibm.com/xmlns/prod/websphere/j2ca/sap) except in the following circumstance. If you are adding the business object to an existing module and the module already includes that business object (from an earlier run of the J2C Bean wizard), change the namespace value.

   For example, you could change the namespace to http://www.ibm.com/xmlns/prod/websphere/j2ca/sap1.
4. To indicate where the business object information should be stored, type the path to the location in the **Folder** field. This is an optional step.
5. If you want to continue to process a BAPI even if the BAPI return object contains errors, select the **Ignore errors in BAPI Return object** check box.
6. Click **Finish**.

Results

You specified a name for the top-level business object and selected an operation for the BAPIs. You also established the order of processing for the BAPIs. The Service Generation and Deployment Configuration window is displayed.

What to do next

Generate a deployable module that includes the adapter and the business objects.

Setting deployment properties and generating artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **Project name** field, select or create a new project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the New Source Project Creation window, select **Java project**.

c. In the Create a new project window, type a name for the project. For example, MyApdapterOutbound.
d. Accept the default values for the other fields.
e. Click **Finish**.

2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new package:
     a. Click **New**.
     b. In the Java Package window, click **Java project**.
     c. In the Create a new project window, type a name for the package. For example, myadapteroutboundpkg.
     d. Accept the default values for the other fields.
     e. Click **Finish**.
3. In the **Interface name** field, specify the interface name you want to use for your business objects. For example, MyAdapterOutboundInterface. The implementation name is automatically generated by suffixing ″Impl″ to the interface name and the name is displayed in the **Implementation name** field. For example, MyAdapterOutboundInterfaceImpl.
4. Optional: Select the **Enable generate Command Bean** check box, to generate a command bean for every operation that you have selected. If you create command bean, you need to specify the command bean name as well as the input and output names.
5. In the Connection properties area, specify how you want the adapter to connect to the database.
   - To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.
   - To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.
6. If you select **Managed Connection (recommended)** check box, specify how you want the adapter to specify the connection properties.
   - To select an existing name, click **Browse**.
   - Otherwise, create a new name.
     a. Click **New**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field. For example, `com/eis/AdapterCF`.
     d. Click **Finish**.
7. If you select the **Non-managed Connection** check box, the Database system connection information area is expanded to show the connection information. Review the connection information and change the values if required. You can change the values to specify a different user name and password at run time. You can also connect to an alternate database, although the schema names must be the same in both databases. The format of the connection properties is database-specific.
8. Optional: Specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties.

9. When you are finished setting properties, click **Finish**.

Results

The new project is added to the Enterprise Explorer perspective. The module is created in the project and artifacts are generated.

The generated artifacts allow you to build an enterprise application that accesses the EIS. You can use the J2C Bean and Java data bindings directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to the Rational Application Developer documentation.

Generating the EJB or JSP project

After you create the Java project, use the Web Page, Web Service, or EJB from J2C Java wizard to create the EJB or JSP project.

About this task

Procedure

1. To start the J2C Bean wizard, go to the Enterprise Explorer of Rational Application Developer for WebSphere Software and click **New>Other>J2C> Web Page, Web Service, or EJB from J2C Java Bean**.
2. Click **Next**.
3. In the J2C Java bean selection window, click **Browse**.
4. In the Find J2C Bean window, type the first letter of the implementation name generated earlier or type the full name and press Enter.
5. Select the implementation name from the list and click **OK**.
6. Click **Next**.
7. In the Deployment Information window, select the Java EE Resource Type as **EJB** or **Sample JSP** and click **Next**.

   Note: In the Deployment Information window, the **Check Configure Resource Adapter Deployment** check box is available for selection only if you have selected the **Non-managed Connection** check box when specifying the deployment settings.
8. If you select **EJB**, the EJB Creation wizard is displayed. This wizard creates the Java project as an EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In the EJB Project window, type an EJB project name. For example, MyAdapterOutboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project name.
     d. In the EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundEJBEAR.
     e. Click **Next**.
     f. In the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.

g. Select the **Generate Deployment Descriptor** check box.
h. Click **Finish** to return to the EJB Project window.
i. Click **Finish** to return to the EJB Creation window.
j. Click **Next**.

Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the EJB and EAR project and return to EJB Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.

k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the EJB EAR Project you created.
l. Click **Finish**.

9. If you select **Sample JSP**, the Simple JSP Creation wizard is displayed. .
   - To select an existing project, click **Browse**. If the desired project name appears in the Project Selection list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In Dynamic Web Project window, type a project name. For example, MyAdapterOutboundWEB.
     c. In EAR Membership area, click **New** to create a new ear project name.
     d. In EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundWEBEAR.
     e. Click **Next**.
     f. In the Java EE Module dependencies area of the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the Dynamic Web Project window.
     i. Click **Finish** to return to the Simple JSP Creation window.
     j. Click **Next**.

     Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the WEB and EAR project and return to Simple JSP Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.

     k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the Web EAR Project you created.
     l. Click **Finish**.
10. Export the project as an EAR file for deployment.

Configuring a module for the BAPI result set interface

To configure a module to use the adapter for BAPI result set processing, you use the J2C Bean wizard in Rational Application Developer for WebSphere Software to discover the BAPIs used to create the result set. You then configure the business objects that are generated and create a deployable module.

Selecting business objects and services for BAPI result set processing

To specify which BAPI functions you want to use and which data you want to process, you provide information in the J2C Bean wizard.

Before you begin

Make sure you have set the connection properties for the J2C Bean wizard.

About this task

Specify search criteria that the J2C Bean wizard uses to discover BAPI functions on the SAP server. The J2C Bean wizard returns a list of BAPI functions that meet the search criteria.

To specify the search criteria and select the BAPI functions, use the following procedure.

Procedure

1. In the Object Discovery and Selection window, indicate the BAPIs you want to work with.
   a. Click **RFC** to enable the filter button.
   b. Click the filter button.

      Note: Instead of using the filter capability, you can expand **RFC** and select the function from the list, or you can expand **BOR**, expand the functional grouping (for example, **Cross-Application Components**), and select the BAPI. Then skip ahead to step 4.
2. From the Filter Properties window, specify information about the BAPIs:
   a. Select **Discover objects by name** or **Discover objects by description** from the **Object attribute to use for discovery** list.
   b. Type a search string (for example, BAPI_CUSTOMER*) representing the BAPI you want to call.

      This is the name of the BAPI in SAP plus an asterisk as a wild card character to indicate that you want a list of all SAP application components that start with the phrase BAPI_CUSTOMER.
   c. Indicate the number of functions you want returned by changing the value in the **Maximum number of objects to retrieve** field or by accepting the default value.
   d. Click **OK**.
3. Select the BAPIs.
   a. Expand **RFC (filtered)**.
   b. Select two BAPIs–GetList and GetDetail. One BAPI represents the query and one representing the results.
4. Click the arrow button to add the BAPIs to the **Selected objects** list.
5. In the Configuration Properties window, perform the following steps for each BAPI to add it to the list of business objects to be imported:
   a. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.

b. If the BAPI has optional parameters associated with it, select the **Select optional parameters to include as child objects** check box, expand **Optional parameters**, and select the type of parameters (import, export, or table) that you want to work with.

   By default, the J2C Bean wizard generates all the parameters required for the selected BAPI, so select this check box and then clear the check boxes for any parameters you do not want to include in your business object.

   For example, if you are adding the ChangeFromData BAPI, you have the option of adding the following parameters:

   PI_DIVISION

   PI_DISTR_CHAN

   Refer to the SAP documentation for a list and description of the optional parameters.

c. Click **OK** to add the BAPI to the list of business objects to be imported.

If you want to remove an object from the list, select the object name and click the left arrow.

6. Click **Next**.

Results

The J2C Bean wizard has returned the functions that match the search criteria, and you have selected the functions you want to work with. The Configure Composite Properties window is displayed.

What to do next

Provide information about the business object (such as the name of the top-level object and associated operation).

Configuring BAPI result set selected objects

To configure a BAPI result set business object, you specify information about the object (such as the name of the object and an indication of which BAPI is used as the query).

Before you begin

Make sure you have selected and imported the BAPI functions.

About this task

To configure the business object, use the following procedure.

Procedure

1. In the Configure Composite Properties window, select a name for the top-level business object.
2. Specify which BAPI is used as the query and select a property that forms the parent-child relationship between BAPIs:
   a. Confirm that the correct BAPI is listed in the **Query BAPI** field. If it is not, select the other BAPI from the list.
   b. Click **Add**.
   c. To display all the properties associated with the first BAPI, click **Select**.
   d. Select the property that you will use to form the parent-child relationship and click **OK**.

e. To display all the properties associated with the second BAPI, click **Select**.
f. Select the property that you will use to form the parent-child relationship and click **OK**.

3. In the **Business object namespace** field, use the default namespace (http://www.ibm.com/xmlns/prod/websphere/j2ca/sap) except in the following circumstance. If you are adding the business object to an existing module and the module already includes that business object (from an earlier run of the J2C Bean wizard), change the namespace value.

   For example, you could change the namespace to http://www.ibm.com/xmlns/prod/websphere/j2ca/sap1.
4. To indicate where the business object information should be stored, type the path to the location in the **Folder** field. This is an optional step.
5. If you want to continue to process a BAPI even if the BAPI return object contains errors, select the **Ignore errors in BAPI Return object** check box.
6. Click **Finish**.

Results

You specified a name for the top-level business object and established the relationship between the BAPIs. The Service Generation and Deployment Configuration window is displayed.

What to do next

Generate a deployable module that includes the adapter and the business objects.

Setting deployment properties and generating artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **Project name** field, select or create a new project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the New Source Project Creation window, select **Java project**.
     c. In the Create a new project window, type a name for the project. For example, MyApdapterOutbound.
     d. Accept the default values for the other fields.
     e. Click **Finish**.
2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.

- To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
- Otherwise, create a new package:
  a. Click **New**.
  b. In the Java Package window, click **Java project**.
  c. In the Create a new project window, type a name for the package. For example, myadapteroutboundpkg.
  d. Accept the default values for the other fields.
  e. Click **Finish**.

3. In the **Interface name** field, specify the interface name you want to use for your business objects. For example, MyAdapterOutboundInterface. The implementation name is automatically generated by suffixing ″Impl″ to the interface name and the name is displayed in the **Implementation name** field. For example, MyAdapterOutboundInterfaceImpl.
4. Optional: Select the **Enable generate Command Bean** check box, to generate a command bean for every operation that you have selected. If you create command bean, you need to specify the command bean name as well as the input and output names.
5. In the Connection properties area, specify how you want the adapter to connect to the database.
   - To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.
   - To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.
6. If you select **Managed Connection (recommended)** check box, specify how you want the adapter to specify the connection properties.
   - To select an existing name, click **Browse**.
   - Otherwise, create a new name.
     a. Click **New**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field. For example, `com/eis/AdapterCF`.
     d. Click **Finish**.
7. If you select the **Non-managed Connection** check box, the Database system connection information area is expanded to show the connection information. Review the connection information and change the values if required. You can change the values to specify a different user name and password at run time. You can also connect to an alternate database, although the schema names must be the same in both databases. The format of the connection properties is database-specific.
8. Optional: Specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties.
9. When you are finished setting properties, click **Finish**.

Results

The new project is added to the Enterprise Explorer perspective. The module is created in the project and artifacts are generated.

The generated artifacts allow you to build an enterprise application that accesses the EIS. You can use the J2C Bean and Java data bindings directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to the Rational Application Developer documentation.

Generating the EJB or JSP project

After you create the Java project, use the Web Page, Web Service, or EJB from J2C Java wizard to create the EJB or JSP project.

About this task

Procedure

1. To start the J2C Bean wizard, go to the Enterprise Explorer of Rational Application Developer for WebSphere Software and click **New>Other>J2C> Web Page, Web Service, or EJB from J2C Java Bean**.
2. Click **Next**.
3. In the J2C Java bean selection window, click **Browse**.
4. In the Find J2C Bean window, type the first letter of the implementation name generated earlier or type the full name and press Enter.
5. Select the implementation name from the list and click **OK**.
6. Click **Next**.
7. In the Deployment Information window, select the Java EE Resource Type as **EJB** or **Sample JSP** and click **Next**.

   Note: In the Deployment Information window, the **Check Configure Resource Adapter Deployment** check box is available for selection only if you have selected the **Non-managed Connection** check box when specifying the deployment settings.
8. If you select **EJB**, the EJB Creation wizard is displayed. This wizard creates the Java project as an EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In the EJB Project window, type an EJB project name. For example, MyAdapterOutboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project name.
     d. In the EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundEJBEAR.
     e. Click **Next**.
     f. In the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the EJB Project window.
     i. Click **Finish** to return to the EJB Creation window.
     j. Click **Next**.

Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the EJB and EAR project and return to EJB Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.

k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the EJB EAR Project you created.

l. Click **Finish**.

9. If you select **Sample JSP**, the Simple JSP Creation wizard is displayed. .
   - To select an existing project, click **Browse**. If the desired project name appears in the Project Selection list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In Dynamic Web Project window, type a project name. For example, MyAdapterOutboundWEB.
     c. In EAR Membership area, click **New** to create a new ear project name.
     d. In EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundWEBEAR.
     e. Click **Next**.
     f. In the Java EE Module dependencies area of the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the Dynamic Web Project window.
     i. Click **Finish** to return to the Simple JSP Creation window.
     j. Click **Next**.

     Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the WEB and EAR project and return to Simple JSP Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.

     k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the Web EAR Project you created.
     l. Click **Finish**.
10. Export the project as an EAR file for deployment.

Configuring the J2C Java bean and Java data bindings for the ALE interface

To configure a module to use the adapter for ALE outbound processing, you use the J2C Bean wizard in Rational Application Developer for WebSphere Softwareto find an IDoc or set of IDocs. You then configure the business objects that are generated and create a deployable module.

Selecting objects for the ALE interface

To specify the IDoc you want to process, you provide information in the J2C Bean wizard.

About this task

For the ALE interface, you can select IDocs in one of two ways:

- You can specify an IDoc or a set of IDocs by entering search criteria (such as the name of the IDoc) and having the J2C Bean wizard search the SAP system.
- You can enter an IDoc definition file name with the complete path to its location on the file system.

  If you choose to discover IDocs from a file, you must first configure the file. The file is generated from information on the SAP server and is then saved to your local file system.

Whichever method you choose, you can also specify the queue on the SAP server to which IDocs should be delivered.

Discovering IDocs from the system:

Use the **Discover IDocs from System** option to have the J2C Bean wizard search for IDocs based on the criteria you specify.

Before you begin

Make sure you have set the connection properties for the J2C Bean wizard.

About this task

Specify search criteria that the J2C Bean wizard uses to discover IDocs on the SAP server.

Note: The **Discover IDoc From System** choice applies to both the ALE interface and the ALE pass-through IDoc interface.

Procedure

1. In the Object Discovery and Selection window, indicate which IDoc you want to work with.
   a. Expand **ALE**.
   b. Click **Discover IDoc From System** to enable the filter button.
   c. Click the filter button.

      Note: Instead of using the filter button, you can expand **Discover IDoc From System** and select the IDoc from the list. You then skip ahead to step 4 on page 90.
2. From the Filter Properties window, specify information about the IDoc or IDocs:
   a. Select **Discover objects by name** or **Discover objects by description** from the **Object attribute to use for discovery** list.
   b. Type a search string (for example, `ALEREQ*`) representing the IDoc you want to call.

      This is the name of the IDoc in SAP plus an asterisk as a wild card character to indicate that you want a list of all IDocs that start with ALEREQ.
   c. Select **Basic IDocs** or **Extension IDocs** from the **IDoc type to use for discovery** field.
   d. Indicate the number of functions you want returned by changing the value in the **Maximum number of objects to retrieve** field or by accepting the default value.

e. Click **OK**.

3. Select the IDoc or IDocs.
   a. Expand **Discover IDoc From System (filtered)**.
   b. Click the IDoc you want to use. If you are working with multiple IDocs, click the names of all the IDocs.
4. Click the arrow button to add the IDoc or IDocs to the **Selected objects** list.
5. In the Configuration Parameters window, perform the following tasks to add the IDoc to the list of business objects to be imported.
   a. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   b. If you want to have IDocs sent to a queue on the SAP server, click **Use qRFC to serialize outbound data using a queue**, and then select the queue from the **Select the queue name** list.
   c. In the **IDoc release version** field, specify the SAP release number to identify the IDoc type you want the J2C Bean wizard to use for creating business objects.
   d. Click **OK**.
6. Click **Next**.

Results

The J2C Bean wizard has returned an IDoc or a list of IDocs, and you have selected the ones you want to work with. You see the Configure Composite Properties window.

Discovering IDocs from a file:

To select IDocs from a file, you must first configure an IDoc definition file based on information on the SAP server. You then specify, in the J2C Bean wizard, the path to the file on your local system.

Before you begin

You must have created an IDoc definition file.

Note: If you are using **Discover IDoc From System**, do not complete the following steps. The IDoc definition file is needed only if you are using **Discover IDoc From File**.

About this task

Specify the IDoc definition file that the J2C Bean wizard uses to discover the IDoc.

Procedure

1. In the Object Discovery and Selection window, indicate which IDoc you want to work with.
   a. Expand **ALE**.
   b. Click **Discover IDoc From File** to enable the filter button.
   c. Click the filter button.

Note: Instead of using the filter button, you can expand **Discover IDoc From File** and select the IDoc definition file. You then skip ahead to step 4.

2. From the Filter Properties window, specify the location of the IDoc definition file.
   a. Click **Browse** to navigate to the IDoc definition file, or type the path to the file.
   b. After you type or select the file, click **OK**.
3. Select the IDoc or IDocs.
   a. Expand **Discover IDoc From File (filtered)**.

      The IDoc definition file is displayed.
   b. Click the IDoc definition file.
4. Click the arrow button to add it to the **Selected objects** list.
5. In the Configuration Parameters window, perform the following tasks:
   a. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   b. If you want to have IDocs sent to a queue on the SAP server, click **Use qRFC to serialize outbound data with a queue**, and then select the queue from the **Select the queue name** list.
   c. In the **IDoc release version** field, specify the SAP release number to identify the IDoc type you want the J2C Bean wizard to use for creating business objects.
   d. Click **OK**.
6. Click **Next**.

Results

The J2C Bean wizard has returned an IDoc or a list of IDocs associated with the IDoc definition file. You see the Configure Composite Properties window.

Configuring the Java data bindings for the ALE interface

To configure the business object, you specify information about the object (such as the name of the folder where the object will be stored).

Before you begin

Make sure you have selected and imported the ALE IDoc.

About this task

Note: This task does not apply to business objects generated with the ALE pass-through IDoc interface.

To configure the business object, use the following procedure.

Procedure

1. In the **Business object namespace** field, use the default namespace (http://www.ibm.com/xmlns/prod/websphere/j2ca/sap) except in the following circumstance. If you are adding the business object to an existing module and the module already includes that business object (from an earlier run of the J2C Bean wizard), change the namespace value.

For example, you could change the namespace to http://www.ibm.com/xmlns/prod/websphere/j2ca/sap1.

2. To indicate where the business object information should be stored, type the path to the location in the **Folder** field. This is an optional step.
3. Click **Next**.

Results

You have optionally specified a location where the object is stored and changed the namespace. The Service Generation and Deployment Configuration window is displayed.

What to do next

Generate a deployable module that includes the adapter and the business objects.

Setting deployment properties and generating artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **Project name** field, select or create a new project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the New Source Project Creation window, select **Java project**.
     c. In the Create a new project window, type a name for the project. For example, MyApdapterOutbound.
     d. Accept the default values for the other fields.
     e. Click **Finish**.
2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new package:
     a. Click **New**.
     b. In the Java Package window, click **Java project**.
     c. In the Create a new project window, type a name for the package. For example, myadapteroutboundpkg.
     d. Accept the default values for the other fields.
     e. Click **Finish**.

3. In the **Interface name** field, specify the interface name you want to use for your business objects. For example, MyAdapterOutboundInterface. The implementation name is automatically generated by suffixing ″Impl″ to the interface name and the name is displayed in the **Implementation name** field. For example, MyAdapterOutboundInterfaceImpl.
4. Optional: Select the **Enable generate Command Bean** check box, to generate a command bean for every operation that you have selected. If you create command bean, you need to specify the command bean name as well as the input and output names.
5. In the Connection properties area, specify how you want the adapter to connect to the database.
   - To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.
   - To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.
6. If you select **Managed Connection (recommended)** check box, specify how you want the adapter to specify the connection properties.
   - To select an existing name, click **Browse**.
   - Otherwise, create a new name.
     a. Click **New**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field. For example, `com/eis/AdapterCF`.
     d. Click **Finish**.
7. If you select the **Non-managed Connection** check box, the Database system connection information area is expanded to show the connection information. Review the connection information and change the values if required. You can change the values to specify a different user name and password at run time. You can also connect to an alternate database, although the schema names must be the same in both databases. The format of the connection properties is database-specific.
8. Optional: Specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties.
9. When you are finished setting properties, click **Finish**.

Results

The new project is added to the Enterprise Explorer perspective. The module is created in the project and artifacts are generated.

The generated artifacts allow you to build an enterprise application that accesses the EIS. You can use the J2C Bean and Java data bindings directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to the Rational Application Developer documentation.

Generating the EJB or JSP project

After you create the Java project, use the Web Page, Web Service, or EJB from J2C Java wizard to create the EJB or JSP project.

About this task

Procedure

1. To start the J2C Bean wizard, go to the Enterprise Explorer of Rational Application Developer for WebSphere Software and click **New>Other>J2C> Web Page, Web Service, or EJB from J2C Java Bean**.
2. Click **Next**.
3. In the J2C Java bean selection window, click **Browse**.
4. In the Find J2C Bean window, type the first letter of the implementation name generated earlier or type the full name and press Enter.
5. Select the implementation name from the list and click **OK**.
6. Click **Next**.
7. In the Deployment Information window, select the Java EE Resource Type as **EJB** or **Sample JSP** and click **Next**.

   Note: In the Deployment Information window, the **Check Configure Resource Adapter Deployment** check box is available for selection only if you have selected the **Non-managed Connection** check box when specifying the deployment settings.
8. If you select **EJB**, the EJB Creation wizard is displayed. This wizard creates the Java project as an EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In the EJB Project window, type an EJB project name. For example, MyAdapterOutboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project name.
     d. In the EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundEJBEAR.
     e. Click **Next**.
     f. In the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the EJB Project window.
     i. Click **Finish** to return to the EJB Creation window.
     j. Click **Next**.

        Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the EJB and EAR project and return to EJB Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.
     k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the EJB EAR Project you created.
     l. Click **Finish**.
9. If you select **Sample JSP**, the Simple JSP Creation wizard is displayed. .

- To select an existing project, click **Browse**. If the desired project name appears in the Project Selection list, select its name.
- Otherwise, create a new project.
  a. Click **New**.
  b. In Dynamic Web Project window, type a project name. For example, MyAdapterOutboundWEB.
  c. In EAR Membership area, click **New** to create a new ear project name.
  d. In EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundWEBEAR.
  e. Click **Next**.
  f. In the Java EE Module dependencies area of the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
  g. Select the **Generate Deployment Descriptor** check box.
  h. Click **Finish** to return to the Dynamic Web Project window.
  i. Click **Finish** to return to the Simple JSP Creation window.
  j. Click **Next**.

     Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the WEB and EAR project and return to Simple JSP Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.
  k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the Web EAR Project you created.
  l. Click **Finish**.

10. Export the project as an EAR file for deployment.

Configuring a module for ALE pass-through IDoc outbound processing

To configure a module to use the adapter for ALE outbound processing, you use the J2C Bean wizard in Rational Application Developer for WebSphere Software to find an IDoc or set of IDocs. You then configure the business objects that are generated and create a deployable module.

Selecting business objects and services for ALE pass-through IDoc outbound processing

To specify the IDoc you want to process, you provide information in the J2C Bean wizard.

About this task

For the ALE pass-through IDoc interface, you can specify IDocs from a system or from a file, but the most likely reason for using the pass-through IDoc interface is to use a generic IDoc.

- When you select a generic IDoc, you create one business-object definition that can apply to any IDoc at run time. This selection is helpful if you are processing many IDocs and do not want to create a separate business-object definition for each one.

- If specify an IDoc from a system or file, you select a specific IDoc (for example, ORDERS05) during configuration. However, you can use a different IDoc during run time when you send the request to the SAP server.

Whichever method you choose, you can also specify the queue on the SAP server to which IDocs should be delivered.

Procedure

1. In the Object Discovery and Selection window, indicate that you want to select a generic IDoc.
   a. Expand **ALE**.
   b. Click **Generic IDoc** .
2. Click the arrow button to add the generic IDoc to the **Selected objects** list.
3. When the Configuration Parameters window is displayed, indicate whether you want to have IDocs sent to a queue on the SAP server:
   - If you do not want to send IDocs to a queue, click **Cancel.**
   - If you want to send IDocs to a queue, perform the following steps:
     a. Click **Use qRFC to serialize outbound data using a queue**.
     b. Select a queue from the **Select the queue name** list.
     c. Click **OK**.
4. Click **Next**.

Results

You have selected a generic IDoc.

What to do next

Set deployment properties and generate a module.

Setting deployment properties and generating artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **Project name** field, select or create a new project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the New Source Project Creation window, select **Java project**.
     c. In the Create a new project window, type a name for the project. For example, MyApdapterOutbound.
     d. Accept the default values for the other fields.

e. Click **Finish**.

2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new package:
     a. Click **New**.
     b. In the Java Package window, click **Java project**.
     c. In the Create a new project window, type a name for the package. For example, myadapteroutboundpkg.
     d. Accept the default values for the other fields.
     e. Click **Finish**.
3. In the **Interface name** field, specify the interface name you want to use for your business objects. For example, MyAdapterOutboundInterface. The implementation name is automatically generated by suffixing ″Impl″ to the interface name and the name is displayed in the **Implementation name** field. For example, MyAdapterOutboundInterfaceImpl.
4. Optional: Select the **Enable generate Command Bean** check box, to generate a command bean for every operation that you have selected. If you create command bean, you need to specify the command bean name as well as the input and output names.
5. In the Connection properties area, specify how you want the adapter to connect to the database.
   - To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.
   - To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.
6. If you select **Managed Connection (recommended)** check box, specify how you want the adapter to specify the connection properties.
   - To select an existing name, click **Browse**.
   - Otherwise, create a new name.
     a. Click **New**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field. For example, `com/eis/AdapterCF`.
     d. Click **Finish**.
7. If you select the **Non-managed Connection** check box, the Database system connection information area is expanded to show the connection information. Review the connection information and change the values if required. You can change the values to specify a different user name and password at run time. You can also connect to an alternate database, although the schema names must be the same in both databases. The format of the connection properties is database-specific.
8. Optional: Specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties.
9. When you are finished setting properties, click **Finish**.

Results

The new project is added to the Enterprise Explorer perspective. The module is created in the project and artifacts are generated.

The generated artifacts allow you to build an enterprise application that accesses the EIS. You can use the J2C Bean and Java data bindings directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to the Rational Application Developer documentation.

Generating the EJB or JSP project

After you create the Java project, use the Web Page, Web Service, or EJB from J2C Java wizard to create the EJB or JSP project.

About this task

Procedure

1. To start the J2C Bean wizard, go to the Enterprise Explorer of Rational Application Developer for WebSphere Software and click **New>Other>J2C> Web Page, Web Service, or EJB from J2C Java Bean**.
2. Click **Next**.
3. In the J2C Java bean selection window, click **Browse**.
4. In the Find J2C Bean window, type the first letter of the implementation name generated earlier or type the full name and press Enter.
5. Select the implementation name from the list and click **OK**.
6. Click **Next**.
7. In the Deployment Information window, select the Java EE Resource Type as **EJB** or **Sample JSP** and click **Next**.

   Note: In the Deployment Information window, the **Check Configure Resource Adapter Deployment** check box is available for selection only if you have selected the **Non-managed Connection** check box when specifying the deployment settings.
8. If you select **EJB**, the EJB Creation wizard is displayed. This wizard creates the Java project as an EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In the EJB Project window, type an EJB project name. For example, MyAdapterOutboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project name.
     d. In the EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundEJBEAR.
     e. Click **Next**.
     f. In the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the EJB Project window.
     i. Click **Finish** to return to the EJB Creation window.

j. Click **Next**.

Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the EJB and EAR project and return to EJB Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.

k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the EJB EAR Project you created.

l. Click **Finish**.

9. If you select **Sample JSP**, the Simple JSP Creation wizard is displayed. .
   - To select an existing project, click **Browse**. If the desired project name appears in the Project Selection list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In Dynamic Web Project window, type a project name. For example, MyAdapterOutboundWEB.
     c. In EAR Membership area, click **New** to create a new ear project name.
     d. In EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundWEBEAR.
     e. Click **Next**.
     f. In the Java EE Module dependencies area of the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the Dynamic Web Project window.
     i. Click **Finish** to return to the Simple JSP Creation window.
     j. Click **Next**.

     Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the WEB and EAR project and return to Simple JSP Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.

     k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the Web EAR Project you created.

     l. Click **Finish**.
10. Export the project as an EAR file for deployment.

Configuring a module for Query interface for SAP Software processing

To configure a module to use the adapter for Query interface for SAP Software outbound processing, you use the J2C Bean wizard in Rational Application Developer for WebSphere Software to find data in an SAP table or a set of tables. You then configure the business objects that are generated and create a deployable module.

Selecting business objects and services

To specify which data you want to query, you provide information in the J2C Bean wizard.

Before you begin

Make sure you have set the connection properties for the J2C Bean wizard.

About this task

Specify search criteria that the J2C Bean wizard uses to query data on the SAP server. The J2C Bean wizard returns the data that meets the search criteria.

You can use the discovered tables to generate individual objects (objects that have no relationship to each other) or to generate objects that have a hierarchical structure.

- If you are generating individual objects, you can import one or more objects from the list of discovered tables at the same time.
- If you are generating hierarchical objects, you must import the parent tables first and then import the child tables.

  When you configure the child tables for import, you can select the parent table you imported earlier as its parent. Repeat this process to add more tables to the hierarchical structure. A hierarchical object with three levels, for example, requires three separate imports to establish the parent-child relationship.

To specify the search criteria, use the following procedure.

Procedure

1. In the Object Discovery and Selection window, indicate which table or tables you want to work with.
   a. Click **QISS** to enable the filter button.
   b. Click the filter button.

      Note: Instead of using the filter capability, you can expand **QISS** and select the table from the list. Then skip ahead to step 4.
2. From the Filter Properties window, specify information about the table.
   a. Select **Discover objects by name** or **Discover objects by description** from the **Object attribute to use for discovery** list.
   b. Type a search string (for example, `KN*`) representing the table.

      This is the name of the table in SAP plus an asterisk as a wild card character to indicate that you want a list of all SAP application components that start with KN.
   c. Indicate the number of objects you want returned by changing the value in the **Maximum number of objects to retrieve** field or by accepting the default value.
   d. Click **OK**.
3. Select the table objects.
   a. Expand **QISS (filtered)**.
   b. Click the table object you want to use.
4. Click the arrow button to add the table object to the **Selected objects** list.
5. In the Configuration Properties for *table* window, provide information about the table:

a. The **Add a WHERE clause** field specifies the primary key to the table. A default value is provided. Change this value if you want to use a different primary key.
b. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
c. Indicate which columns you want included in the query.

   Note: There are many columns and, by default, all the columns are selected. You can clear the check from those columns you do not want included, or if you want to select only a few columns, you can use the **Select or unselect all columns** check box.

   For example, if you want only two columns, clear **Select or unselect all columns** to remove the check from all columns, and then select the two columns you want.
d. Click **OK**

6. To include another table in the query, perform the following tasks:
   a. Click **QISS** to enable the filter button.
   b. Click the filter button.

      Note: Instead of using the filter capability, you can expand **QISS** and select the table from the list.
7. From the Filter Properties window, specify information about the table.
   a. Select **Discover objects by name** or **Discover objects by description** from the **Object attribute to use for discovery** list.
   b. Type a search string (for example, `ADRC`) representing the table.
   c. Indicate the number of functions you want returned by changing the value in the **Maximum number of objects to retrieve** field or by accepting the default value.
   d. Click **OK**.
8. Select the table objects.
   a. Expand **QISS (filtered)**.
   b. Click the second table object.
   c. Click the arrow button to add the table object to the **Selected objects** list.
9. In the Configuration Properties *table* window, provide information about the table:
   a. The **Add a WHERE clause** field specifies the primary key to the table. A default value is provided. Change this value if you want to use a different primary key.
   b. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   c. Associate this table with the one you previously added (KNA1 in the example) by selecting that table from the **Select a parent table** section of the window.
   d. Under **Map the primary key columns to the parent-table foreign key reference columns**, select a value to link the tables.

      For example, you might select **ADRNR** for **ADDRNUMBER**.
   e. Indicate which columns you want included in the query.
   f. Click **OK**

10. Click **Next**.

Results

The J2C Bean wizard returns the data that matches the search criteria.

What to do next

From the Configure Composite Properties window, optionally specify a namespace and directory to which the generated business object will be stored.

Configuring the selected objects

To configure the object, you specify information about where the object should be stored.

Before you begin

Make sure you have selected and imported the business object.

About this task

To configure the business object, use the following procedure.

Procedure

1. In the **Business object namespace** field, use the default namespace (http://www.ibm.com/xmlns/prod/websphere/j2ca/sap) except in the following circumstance. If you are adding the business object to an existing module and the module already includes that business object (from an earlier run of the J2C Bean wizard), change the namespace value.

   For example, you could change the namespace to http://www.ibm.com/xmlns/prod/websphere/j2ca/sap1.
2. To indicate where the business object information should be stored, type the path to the location in the **Folder** field. This is an optional step.
3. Click **Next** to continue to the Service Generation and Deployment Configuration window, except in the following circumstance:

   If you created a custom function module on the SAP server (per SAP note 758278) to avoid the `Error in ASSIGN statement in the program SAPLSDTX` exception, indicate the name of the module:

   a. Click **Advanced**.
   b. In the **Custom retrieve function name field**, type the name of the function.
   c. Click **Next**.

Results

You have made changes to the default settings (for example, changing the namespace), or you have accepted all the default settings. The Service Generation and Deployment Configuration window is displayed.

What to do next

Generate a deployable module that includes the adapter and the business objects.

Setting deployment properties and generating artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **Project name** field, select or create a new project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the New Source Project Creation window, select **Java project**.
     c. In the Create a new project window, type a name for the project. For example, MyApdapterOutbound.
     d. Accept the default values for the other fields.
     e. Click **Finish**.
2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new package:
     a. Click **New**.
     b. In the Java Package window, click **Java project**.
     c. In the Create a new project window, type a name for the package. For example, myadapteroutboundpkg.
     d. Accept the default values for the other fields.
     e. Click **Finish**.
3. In the **Interface name** field, specify the interface name you want to use for your business objects. For example, MyAdapterOutboundInterface. The implementation name is automatically generated by suffixing ″Impl″ to the interface name and the name is displayed in the **Implementation name** field. For example, MyAdapterOutboundInterfaceImpl.
4. Optional: Select the **Enable generate Command Bean** check box, to generate a command bean for every operation that you have selected. If you create command bean, you need to specify the command bean name as well as the input and output names.
5. In the Connection properties area, specify how you want the adapter to connect to the database.
   - To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.
   - To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.

6. If you select **Managed Connection (recommended)** check box, specify how you want the adapter to specify the connection properties.
   - To select an existing name, click **Browse**.
   - Otherwise, create a new name.
     a. Click **New**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field. For example, `com/eis/AdapterCF`.
     d. Click **Finish**.
7. If you select the **Non-managed Connection** check box, the Database system connection information area is expanded to show the connection information. Review the connection information and change the values if required. You can change the values to specify a different user name and password at run time. You can also connect to an alternate database, although the schema names must be the same in both databases. The format of the connection properties is database-specific.
8. Optional: Specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties.
9. When you are finished setting properties, click **Finish**.

Results

The new project is added to the Enterprise Explorer perspective. The module is created in the project and artifacts are generated.

The generated artifacts allow you to build an enterprise application that accesses the EIS. You can use the J2C Bean and Java data bindings directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to the Rational Application Developer documentation.

Generating the EJB or JSP project

After you create the Java project, use the Web Page, Web Service, or EJB from J2C Java wizard to create the EJB or JSP project.

About this task

Procedure

1. To start the J2C Bean wizard, go to the Enterprise Explorer of Rational Application Developer for WebSphere Software and click **New>Other>J2C> Web Page, Web Service, or EJB from J2C Java Bean**.
2. Click **Next**.
3. In the J2C Java bean selection window, click **Browse**.
4. In the Find J2C Bean window, type the first letter of the implementation name generated earlier or type the full name and press Enter.
5. Select the implementation name from the list and click **OK**.
6. Click **Next**.
7. In the Deployment Information window, select the Java EE Resource Type as **EJB** or **Sample JSP** and click **Next**.

Note: In the Deployment Information window, the **Check Configure Resource Adapter Deployment** check box is available for selection only if you have selected the **Non-managed Connection** check box when specifying the deployment settings.

8. If you select **EJB**, the EJB Creation wizard is displayed. This wizard creates the Java project as an EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In the EJB Project window, type an EJB project name. For example, MyAdapterOutboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project name.
     d. In the EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundEJBEAR.
     e. Click **Next**.
     f. In the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the EJB Project window.
     i. Click **Finish** to return to the EJB Creation window.
     j. Click **Next**.

     Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the EJB and EAR project and return to EJB Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.
     k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the EJB EAR Project you created.
     l. Click **Finish**.
9. If you select **Sample JSP**, the Simple JSP Creation wizard is displayed. .
   - To select an existing project, click **Browse**. If the desired project name appears in the Project Selection list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In Dynamic Web Project window, type a project name. For example, MyAdapterOutboundWEB.
     c. In EAR Membership area, click **New** to create a new ear project name.
     d. In EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundWEBEAR.
     e. Click **Next**.
     f. In the Java EE Module dependencies area of the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the Dynamic Web Project window.

i. Click **Finish** to return to the Simple JSP Creation window.
j. Click **Next**.

   Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the WEB and EAR project and return to Simple JSP Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.
k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the Web EAR Project you created.
l. Click **Finish**.

10. Export the project as an EAR file for deployment.

Configuring a module for Advanced event processing - outbound

To configure a module to use the adapter for Advanced event processing, you use the J2C Bean wizard in Rational Application Developer for WebSphere Software to discover IDocs on the SAP server. You then configure the business objects that are generated and create a deployable module. To use the Advanced event processing interface, you must first add the adapter-supplied transport files to the SAP server.

Selecting business objects and services for Advanced event (outbound) processing

To specify which function you want to process, you provide information in the J2C Bean wizard.

Before you begin

Make sure you have set the connection properties for the J2C Bean wizard.

About this task

Specify search criteria that the J2C Bean wizard uses to discover functions on the SAP server. The J2C Bean wizard returns a list of functions that meet the search criteria.

To specify the search criteria and select one or more functions, use the following procedure.

Procedure

1. In the Object Discovery and Selection window, indicate which IDoc you want to work with.
   a. Expand **AEP**.
   b. Click **Discover IDoc From System** to enable the filter button.
   c. Click the filter button.

      Note: Instead of using the filter button, you can expand **Discover IDoc From System** and select the IDoc from the list. Then skip ahead to step 4 on page 107.
2. From the Filter Properties window, specify information about the IDoc or IDocs:

a. Select **Discover objects by name** or **Discover objects by description** from the **Object attribute to use for discovery** list.
b. Type a search string representing the IDoc you want to call.
c. Select **Basic IDocs** or **Extension IDocs** from the **IDoc type to use for discovery** field.
d. Indicate the number of functions you want returned by changing the value in the **Maximum number of objects to retrieve** field or by accepting the default value.
e. Click **OK**.

3. Select the IDoc or IDocs.
   a. Expand **Discover IDoc From System (filtered)**.
   b. Click the IDoc you want to use. If you are working with multiple IDocs, click the names of all the IDocs.
4. Click the arrow button to add the IDoc or IDocs to the **Selected objects** list.
5. In the Configuration Parameters window, perform the following steps to add the IDoc to the list of business objects to be imported.
   a. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   b. In the **IDoc release version** field, specify the SAP release number to identify the IDoc type you want the J2C Bean wizard to use for creating business objects.
   c. Expand the IDoc name and select one or more nodes to be used as the primary key, or leave the default values selected.
   d. Click **OK**.
6. Click **Next**.

Results

The J2C Bean wizard has returned a function or list of functions that match the search criteria, and you have selected the function or functions you want to work with.

What to do next

From the Configure Composite Properties window, select an operation for the IDoc and an ABAP function module for that operation. Optionally specify a namespace and directory to which the generated business object will be stored.

Configuring the selected objects

To configure the object, you associate an operation with the IDoc and associate an ABAP function module with the selected operation.

Before you begin

Make sure you have selected and imported the function.

About this task

To configure the business object, use the following procedure.

Procedure

1. In the Configure Composite Properties window, click an IDoc from the **IDoc to configure** list.

   If you are configuring only one IDoc, this step is not necessary.
2. Click **Add** in the Service operations for selected IDoc section of the window.
3. Select an operation (for example, **Retrieve**), and click **OK**.
4. In the **ABAP function module name for selected operations** field, type the name of the ABAP function module to associate with this operation.

   Note: The ABAP function module must have been created and must exist on the SAP server.
5. If you are working with multiple IDocs, repeat the previous four steps for each IDoc.
6. In the **Business object namespace** field, use the default namespace (http://www.ibm.com/xmlns/prod/websphere/j2ca/sap) except in the following circumstance. If you are adding the business object to an existing module and the module already includes that business object (from an earlier run of the J2C Bean wizard), change the namespace value.

   For example, you could change the namespace to http://www.ibm.com/xmlns/prod/websphere/j2ca/sap1.
7. To indicate where the business object information should be stored, type the path to the location in the **Folder** field. This is an optional step.
8. Click **Finish**.

Results

You have associated an operation with each IDoc and have associated an ABAP function module with each operation. The Service Generation and Deployment Configuration window is displayed.

What to do next

Generate a deployable module that includes the adapter and the business objects.

Setting deployment properties and generating artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **Project name** field, select or create a new project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the New Source Project Creation window, select **Java project**.

c. In the Create a new project window, type a name for the project. For example, MyApdapterOutbound.
d. Accept the default values for the other fields.
e. Click **Finish**.

2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new package:
     a. Click **New**.
     b. In the Java Package window, click **Java project**.
     c. In the Create a new project window, type a name for the package. For example, myadapteroutboundpkg.
     d. Accept the default values for the other fields.
     e. Click **Finish**.
3. In the **Interface name** field, specify the interface name you want to use for your business objects. For example, MyAdapterOutboundInterface. The implementation name is automatically generated by suffixing ″Impl″ to the interface name and the name is displayed in the **Implementation name** field. For example, MyAdapterOutboundInterfaceImpl.
4. Optional: Select the **Enable generate Command Bean** check box, to generate a command bean for every operation that you have selected. If you create command bean, you need to specify the command bean name as well as the input and output names.
5. In the Connection properties area, specify how you want the adapter to connect to the database.
   - To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.
   - To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.
6. If you select **Managed Connection (recommended)** check box, specify how you want the adapter to specify the connection properties.
   - To select an existing name, click **Browse**.
   - Otherwise, create a new name.
     a. Click **New**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field. For example, `com/eis/AdapterCF`.
     d. Click **Finish**.
7. If you select the **Non-managed Connection** check box, the Database system connection information area is expanded to show the connection information. Review the connection information and change the values if required. You can change the values to specify a different user name and password at run time. You can also connect to an alternate database, although the schema names must be the same in both databases. The format of the connection properties is database-specific.
8. Optional: Specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties.

9. When you are finished setting properties, click **Finish**.

Results

The new project is added to the Enterprise Explorer perspective. The module is created in the project and artifacts are generated.

The generated artifacts allow you to build an enterprise application that accesses the EIS. You can use the J2C Bean and Java data bindings directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to the Rational Application Developer documentation.

Generating the EJB or JSP project

After you create the Java project, use the Web Page, Web Service, or EJB from J2C Java wizard to create the EJB or JSP project.

About this task

Procedure

1. To start the J2C Bean wizard, go to the Enterprise Explorer of Rational Application Developer for WebSphere Software and click **New>Other>J2C> Web Page, Web Service, or EJB from J2C Java Bean**.
2. Click **Next**.
3. In the J2C Java bean selection window, click **Browse**.
4. In the Find J2C Bean window, type the first letter of the implementation name generated earlier or type the full name and press Enter.
5. Select the implementation name from the list and click **OK**.
6. Click **Next**.
7. In the Deployment Information window, select the Java EE Resource Type as **EJB** or **Sample JSP** and click **Next**.

   Note: In the Deployment Information window, the **Check Configure Resource Adapter Deployment** check box is available for selection only if you have selected the **Non-managed Connection** check box when specifying the deployment settings.
8. If you select **EJB**, the EJB Creation wizard is displayed. This wizard creates the Java project as an EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In the EJB Project window, type an EJB project name. For example, MyAdapterOutboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project name.
     d. In the EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundEJBEAR.
     e. Click **Next**.
     f. In the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.

g. Select the **Generate Deployment Descriptor** check box.
h. Click **Finish** to return to the EJB Project window.
i. Click **Finish** to return to the EJB Creation window.
j. Click **Next**.

Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the EJB and EAR project and return to EJB Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.

k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the EJB EAR Project you created.
l. Click **Finish**.

9. If you select **Sample JSP**, the Simple JSP Creation wizard is displayed. .
   - To select an existing project, click **Browse**. If the desired project name appears in the Project Selection list, select its name.
   - Otherwise, create a new project.
     a. Click **New**.
     b. In Dynamic Web Project window, type a project name. For example, MyAdapterOutboundWEB.
     c. In EAR Membership area, click **New** to create a new ear project name.
     d. In EAR Application Project window, type an EAR project name. For example, MyAdapterOutboundWEBEAR.
     e. Click **Next**.
     f. In the Java EE Module dependencies area of the Enterprise Application window, select the SAP adapter RAR dependency such as CWYAP_SAPAdapter.
     g. Select the **Generate Deployment Descriptor** check box.
     h. Click **Finish** to return to the Dynamic Web Project window.
     i. Click **Finish** to return to the Simple JSP Creation window.
     j. Click **Next**.

     Note: In the EJB Creation window, the **Next** button is available for selection only if you have selected the **Configure Resource Adapter Deployment** check box when specifying the deployment information. After you create the WEB and EAR project and return to Simple JSP Creation window, you can directly click **Finish** to complete the process. The adapter is deployed as stand alone by default.

     k. In Resource Adapter Deployment window, select **Deploy within EAR** and select the Web EAR Project you created.
     l. Click **Finish**.
10. Export the project as an EAR file for deployment.

Configuring the module for inbound processing

To configure a module to use the adapter for inbound processing, use the J2C Bean wizard in Rational Application Developer for WebSphere Software to find and select business objects and services from the SAP server, and to generate business object definitions and related artifacts.

Configuring a module for BAPI inbound processing

To configure a module to use the adapter for BAPI inbound processing, you use the J2C Bean wizard in Rational Application Developer for WebSphere Software to find RFC-enabled functions. You then configure the business objects that are generated and create a deployable module.

Selecting business objects and services for BAPI inbound processing

To specify which function you want to process, you provide information in the J2C Bean wizard.

Before you begin

Make sure you have set the connection properties for the J2C Bean wizard.

About this task

Specify search criteria that the J2C Bean wizard uses to discover functions on the SAP server. The J2C Bean wizard returns a list of functions that meet the search criteria.

To specify the search criteria and select one or more functions, use the following procedure.

Procedure

1. In the Object Discovery and Selection window, indicate which BAPI or set of BAPIs you want to work with.
   a. Click **RFC** to enable the filter button.
   b. Click the filter button.

      Note: Instead of using the filter capability, you can expand **RFC** and select the function from the list, or you can expand **BOR**, expand the functional grouping (for example, **Cross-Application Components**), and select the BAPI. Then skip ahead to step 4.
2. From the Filter Properties window, specify information about the BAPI or BAPIs you want to discover:
   a. Select **Discover objects by name** or **Discover objects by description** from the **Object attribute to use for discovery** list.
   b. Type a search string (for example, BAPI_CUSTOMER*) representing the BAPI you want to call.

      This is the name of the BAPI in SAP plus an asterisk as a wild card character to indicate that you want a list of all SAP application components that start with the phrase BAPI_CUSTOMER.
   c. Indicate the number of functions you want returned by changing the value in the **Maximum number of objects to retrieve** field or by accepting the default value.
   d. Click **OK**.
3. Select the BAPI or BAPIs.
   a. Expand **RFC (filtered)**.
   b. Click the BAPI you want to use. If you are working with multiple BAPIs, click the names of all the BAPIs.
4. Click the arrow button to add the BAPI or BAPIs to the **Selected objects** list.

5. In the Configuration Parameters window, perform the following steps for each BAPI to add it to the list of business objects to be imported:
   a. Optionally select the **Use SAP field name to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   b. If the BAPI has optional parameters associated with it, select the **Select optional parameters to include as child objects** check box, expand **Optional parameters**, and select the type of parameters (import, export, or table) that you want to work with.

      By default, the J2C Bean wizard generates all the parameters required for the selected BAPI, so select this check box and then clear the check boxes for any parameters you do not want to include in your business object.

      For example, if you are adding the ChangeFromData BAPI, you have the option of adding the following parameters:

      PI_DIVISION

      PI_DISTR_CHAN

      Refer to the SAP documentation for a list and description of the optional parameters.
   c. Click **OK** to add the BAPI to the list of business objects to be imported.

   If you want to remove an object from the list, select the object name and click the left arrow.
6. Click **Next**

Results

The J2C Bean wizard has returned the function or list of functions that match the search criteria, and you have selected the function or functions you want to work with. The Configure Composite Properties window is displayed.

What to do next

Provide information about the business object (such as the operation associated with the object and the SAP remote function call type).

Configuring the selected objects

To configure the object, you specify information about the object (such as the operation associated with the object and the type of remote function call).

Before you begin

If you are sending the function call from a queue on the SAP server (which ensures the order in which BAPIs are delivered), make sure you have configured an outbound queue on the SAP server. You also need an ABAP program on the SAP server that delivers the BAPI events to the outbound queue.

About this task

During configuration of the object, you select which type of remote function call you want to make. You can select **Synchronous RFC** (the default) or **Asynchronous Transactional/Queued RFC**.

- Use **Synchronous RFC** when you want to wait for a response from the endpoint. The endpoint must be available when you send the function call from the SAP server to the adapter.

- Use **Asynchronous Transactional/Queued RFC** in the following circumstances:
  - When you are sending a function call from a queue on the SAP server to the adapter
  - When you want the function call to succeed regardless of whether the endpoint is available at the time of the call.

To configure the business object, use the following procedure.

Procedure

1. In the Configure Composite Properties window, select an operation for each BAPI you selected in the previous task.
   - If you are working with one BAPI, select an operation for that BAPI from the **Operations** list.
   - If you are working with multiple BAPIs, select an operation for each BAPI from the list next to the name of the BAPI. Make sure you select one operation for each BAPI.
2. In the **Business object namespace** field, use the default namespace (http://www.ibm.com/xmlns/prod/websphere/j2ca/sap) except in the following circumstance. If you are adding the business object to an existing module and the module already includes that business object (from an earlier run of the J2C Bean wizard), change the namespace value.

   For example, you could change the namespace to http://www.ibm.com/xmlns/prod/websphere/j2ca/sap1.
3. To indicate where the business object information should be stored, type the path to the location in the **Folder** field. This is an optional step.
4. Select the type of remote function call you want to make.

   Note: If you do not select a type of remote function call, the default (**Synchronous RFC**) is used. In Synchronous RFC, the SAP server sends the BAPI and then waits for the response from the endpoint.

   a. Select the arrow next to the **SAP Remote Function Call (RFC) type** list.
   b. Select one of the RFC types:
      - Select **Asynchronous Transactional/Queued RFC** when you are sending the function call from a queue on the SAP server or if you want the call to succeed regardless of whether the receiving system (the endpoint) is available.
        - If the adapter is available, the call succeeds.
        - If the adapter is not available, the SAP server continues to attempt to make the call until the adapter is available. The SAP system ensures that the call is invoked only once. A transaction ID (TID) is associated with the BAPI.
      - Select **Synchronous RFC** (the default) if you want the BAPI sent in a synchronous manner (the SAP server sends the BAPI and then waits for the response from the endpoint). Note that the endpoint must be available when you use **Synchronous RFC**.
5. Click **Next**.

Results

You selected an operation for each BAPI. The Service Generation and Deployment Configuration window is displayed.

What to do next

Generate a deployable module that includes the adapter and the business object.

Setting deployment properties and generating artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **EJB Project name** field, select or create a new EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the EJB Project window, type a project name. For example, MyAdapterInboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project.
     d. In EAR Application Project window, type an EAR project name. For example, MyAdapterInboundEJBEAR
     e. Click **Finish** to return to the EJB Project window.
     f. Click **Finish**.
2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new package:
     a. Click **New**.
     b. In the New Java Package window, type a name for the package. For example, myadapterinboundejbpkg.
     c. Click **Finish**.
3. In the **Stateless Session EJB's local business interface name** field, specify the interface name you want to use for your business object. For example, MyAdapterInboundInterface. The interface name is suffixed with "MDB" and it is displayed automatically in the **Message Driven EJB name** field. For example, MyAdapterInboundInterfaceMDB. Similarly, the interface name is suffixed with "SB" and it is displayed automatically in the **Stateless Session EJB name** field. For example, MyAdapterInboundInterfaceSB.
4. In the Inbound Connection configuration area, specify the JNDI name, which JNDI data source for an existing activation specification in WAS server or you can create it later. For example, `com/eis/AdapterAS`.
5. Click **Finish**.

Configuring a module for ALE inbound processing

To configure a module to use the adapter for ALE inbound processing, you use the J2C Bean wizard in Rational Application Developer for WebSphere Software to find an IDoc or set of IDocs, configure the business objects that are generated, and create a deployable module. If you are going to set up an event recovery table to persist inbound events (to ensure once-only delivery of events), you must also set up a data source.

Selecting business objects and services for ALE inbound processing

To specify the IDoc you want to process, you provide information in the J2C Bean wizard.

About this task

For the ALE interface, you can select IDocs in one of two ways:

- You can specify an IDoc or a set of IDocs by entering search criteria (such as the name of the IDoc) and having the J2C Bean wizard search the SAP system.
- You can enter an IDoc definition file name with the complete path to its location on the file system.

  If you choose to discover IDocs from a file, you must first configure the file. The file is generated from information on the SAP server and is then saved to your local file system.

For the ALE pass-through IDoc interface, you can specify IDocs from a system or from a file, as described in the previous section. Additionally, you can select a generic IDoc.

When you select a generic IDoc, you create one business-object definition that can apply to any IDoc at run time. This selection is helpful if you are processing many IDocs and do not want to create a separate business-object definition for each one.

Discovering IDocs from the system:

Use the **Discover IDocs from System** option to have the J2C Bean wizard search for IDocs based on the criteria you specify.

Before you begin

Make sure you have set the connection properties for the J2C Bean wizard.

About this task

Specify search criteria that the J2C Bean wizard uses to discover IDocs on the SAP server.

Note: The **Discover IDoc From System** choice applies to both the ALE interface and the ALE pass-through IDoc interface.

Procedure

1. In the Object Discovery and Selection window, indicate which IDoc you want to work with.
   a. Expand **ALE**.
   b. Click **Discover IDoc From System** to enable the filter button.

c. Click the filter button.

   Note: Instead of using the filter button, you can expand **Discover IDoc From System** and select the IDoc from the list. You then skip ahead to step 4.

2. From the Filter Properties window, specify information about the IDoc or IDocs:
   a. Select **Discover objects by name** or **Discover objects by description** from the **Object attribute to use for discovery** list.
   b. Type a search string (for example, `ALEREQ*`) representing the IDoc you want to call.

      This is the name of the IDoc in SAP plus an asterisk as a wild card character to indicate that you want a list of all IDocs that start with ALEREQ.
   c. Select **Basic IDocs** or **Extension IDocs** from the **IDoc type to use for discovery** field.
   d. Indicate the number of functions you want returned by changing the value in the **Maximum number of objects to retrieve** field or by accepting the default value.
   e. Click **OK**.
3. Select the IDoc or IDocs.
   a. Expand **Discover IDoc From System (filtered)**.
   b. Click the IDoc you want to use. If you are working with multiple IDocs, click the names of all the IDocs.
4. Click the arrow button to add the IDoc or IDocs to the **Selected objects** list.
5. In the Configuration Parameters window, perform the following tasks to add the IDoc to the list of business objects to be imported.

   Note: If you are using the ALE pass-through IDoc interface, only the **Send an IDoc Packet as one business object** configuration property is available.
   a. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   b. If you are working with an IDoc packet and want to specify that the packet not be split, select the **Send an IDoc Packet as one business object** check box.
   c. If you want to send the IDoc in an unparsed form (so that the client application, rather than the adapter, parses the data), select the **Send an IDoc packet as unparsed data** check box.
   d. In the **IDoc release version** field, specify the SAP release number to identify the IDoc type you want the J2C Bean wizard to use for creating business objects.
   e. Click **OK**.
6. Click **Next**.

Results

The J2C Bean wizard has returned an IDoc or a list of IDocs, and you have selected the ones you want to work with.You see the Configure Composite

Properties window (if you are using the ALE interface) or the Service Generation and Deployment Configuration (if you are using the ALE pass-through IDoc interface).

What to do next

- If you are using the ALE interface, you can optionally specify a namespace and directory to which the generated business object will be stored as described in "Configuring the selected objects" on page 119.
- If you are using the ALE pass-through IDoc interface, you generate a deployable module that includes the adapter and the business objects, as described in "Setting deployment properties and generating artifacts" on page 115.

Discovering IDocs from a file:

To select IDocs from a file, you must first configure an IDoc definition file based on information on the SAP server. You then specify, in the J2C Bean wizard, the path to the file on your local system.

Before you begin

You must have created an IDoc definition file.

Note: If you are using **Discover IDoc From System**, do not complete the following steps. The IDoc definition file is needed only if you are using **Discover IDoc From File**.

About this task

Specify the IDoc definition file that the J2C Bean wizard uses to discover the IDoc.

Note: The **Discover IDoc From File** choice applies to both the ALE interface and the ALE pass-through IDoc interface.

Procedure

1. In the Object Discovery and Selection window, indicate which IDoc you want to work with.
   a. Expand **ALE**.
   b. Click **Discover IDoc From File** to enable the filter button.
   c. Click the filter button.

      Note: Instead of using the filter button, you can expand **Discover IDoc From File** and select the IDoc definition file. You then skip ahead to step 4.
2. From the Filter Properties window, specify the location of the IDoc definition file.
   a. Click **Browse** to navigate to the IDoc definition file, or type the path to the file.
   b. After you type or select the file, click **OK**.
3. Select the IDoc or IDocs.
   a. Expand **Discover IDoc From File (filtered)**.
      The IDoc definition file is displayed.
   b. Click the IDoc definition file.
4. Click the arrow button to add it to the **Selected objects** list.

5. In the Configuration Parameters window, perform the following tasks:

   Note: If you are using the ALE pass-through IDoc interface, only the **Send an IDoc Packet as one business object** configuration property is available.

   a. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   b. If you are working with an IDoc packet and want to specify that the packet not be split, select the **Send an IDoc Packet as one business object** check box.
   c. If you want to send the IDoc in an unparsed form (so that the client application, rather than the adapter, parses the data), select the **Send an IDoc packet as unparsed data** check box.
   d. In the **IDoc release version** field, specify the SAP release number to identify the IDoc type you want the J2C Bean wizard to use for creating business objects.
   e. Click **OK**.
6. Click **Next**.

Results

The J2C Bean wizard has returned an IDoc or a list of IDocs associated with the IDoc definition file. You see the Configure Composite Properties window (if you are using the ALE interface) or the Service Generation and Deployment Configuration (if you are using the ALE pass-through IDoc interface).

What to do next

- If you are using the ALE interface, you can optionally specify a namespace and directory to which the generated business object will be stored as described in "Configuring the Java data bindings for the ALE interface" on page 91.
- If you are using the ALE pass-through IDoc interface, you generate a deployable module that includes the adapter and the business objects, as described in "Setting deployment properties and generating artifacts" on page 115.

Configuring the selected objects

To configure the business object, you specify information about the object (such as the operation associated with the object).

Before you begin

Make sure you have selected and imported the ALE IDoc.

About this task

Note: This task does not apply to business objects generated with the ALE pass-through IDoc interface.

To configure the business object, use the following procedure.

Procedure

1. In the Configure Composite Properties window, click an IDoc from the **IDoc to configure** list.

   If you are configuring only one IDoc, this step is not necessary.

2. Click **Add** in the Service operations for selected IDoc section of the window.
3. Select an operation (for example, **Create**), and click **OK**.
4. From the **IDoc values to identify selected operation** list, select a set of values to associate the IDoc message type, message code, and message function values with the selected service operation.

   At run time, the adapter uses these values to identify the service operation at the endpoint for invocation.

   All the possible combinations of message type, code, and function for the selected IDoc are listed.
5. If you are working with multiple IDocs, repeat the previous four steps for each IDoc.
6. In the **Business object namespace** field, use the default namespace (http://www.ibm.com/xmlns/prod/websphere/j2ca/sap) except in the following circumstance. If you are adding the business object to an existing module and the module already includes that business object (from an earlier run of the J2C Bean wizard), change the namespace value.

   For example, you could change the namespace to http://www.ibm.com/xmlns/prod/websphere/j2ca/sap1.
7. To indicate where the business object information should be stored, type the path to the location in the **Folder** field. This is an optional step.
8. Click **Next**.

Results

You have associated an operation with each IDoc and have selected the combination of message type, code, and function. The Service Generation and Deployment Configuration window is displayed.

What to do next

Generate a deployable module that includes the adapter and the business object.

Setting deployment properties and generating artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **EJB Project name** field, select or create a new EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the EJB Project window, type a project name. For example, MyAdapterInboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project.

d. In EAR Application Project window, type an EAR project name. For example, MyAdapterInboundEJBEAR
e. Click **Finish** to return to the EJB Project window.
f. Click **Finish**.

2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new package:
     a. Click **New**.
     b. In the New Java Package window, type a name for the package. For example, myadapterinboundejbpkg.
     c. Click **Finish**.
3. In the **Stateless Session EJB's local business interface name** field, specify the interface name you want to use for your business object. For example, MyAdapterInboundInterface. The interface name is suffixed with "MDB" and it is displayed automatically in the **Message Driven EJB name** field. For example, MyAdapterInboundInterfaceMDB. Similarly, the interface name is suffixed with "SB" and it is displayed automatically in the **Stateless Session EJB name** field. For example, MyAdapterInboundInterfaceSB.
4. In the Inbound Connection configuration area, specify the JNDI name, which JNDI data source for an existing activation specification in WAS server or you can create it later. For example, `com/eis/AdapterAS`.
5. Click **Finish**.

Configuring a module for ALE pass-through IDoc inbound processing

To configure a module to use the adapter for ALE inbound processing, you use the J2C Bean wizard Rational Application Developer for WebSphere Software in to find an IDoc or set of IDocs, configure the business objects that are generated, and create a deployable module. If you are going to set up an event recovery table to persist inbound events (to ensure once-only delivery of events), you must also set up a data source.

Selecting business objects and services for ALE pass-through IDoc inbound processing

To specify the IDoc you want to process, you provide information in the J2C Bean wizard.

About this task

For the ALE pass-through IDoc interface, you can specify IDocs from a system or from a file, but the most likely reason for using the pass-through IDoc interface is to use a generic IDoc.

When you select a generic IDoc, you create one business-object definition that can apply to any IDoc at run time. This selection is helpful if you are processing many IDocs and do not want to create a separate business-object definition for each one.

Note: You see the **Generic IDoc** choice only if you selected **ALE pass-through IDoc** as the interface on the Discovery Connection window.

Procedure

1. In the Object Discovery and Selection window, indicate that you want to select a generic IDoc.
   a. Expand **ALE**.
   b. Click **Generic IDoc** .
2. Click the arrow button to add the generic IDoc to the **Selected objects** list.
3. When the Configuration Parameters window is displayed, indicate whether you want to have multiple IDocs sent as one packet (instead of sending them as individual business objects):
   - If you do not want to send multiple IDocs as a packet, click **Cancel.**
   - If you want to send multiple IDocs as a packet, click **Send an IDoc packet as one business object** and click **OK**.
4. Click **Next**.

Results

You have selected a generic IDoc.

What to do next

Set deployment properties and generate a module.

Setting deployment properties and generating artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **EJB Project name** field, select or create a new EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the EJB Project window, type a project name. For example, MyAdapterInboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project.
     d. In EAR Application Project window, type an EAR project name. For example, MyAdapterInboundEJBEAR
     e. Click **Finish** to return to the EJB Project window.
     f. Click **Finish**.
2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new package:

a. Click **New**.
b. In the New Java Package window, type a name for the package. For example, myadapterinboundejbpkg.
c. Click **Finish**.

3. In the **Stateless Session EJB's local business interface name** field, specify the interface name you want to use for your business object. For example, MyAdapterInboundInterface. The interface name is suffixed with "MDB" and it is displayed automatically in the **Message Driven EJB name** field. For example, MyAdapterInboundInterfaceMDB. Similarly, the interface name is suffixed with ″SB″ and it is displayed automatically in the **Stateless Session EJB name** field. For example, MyAdapterInboundInterfaceSB.
4. In the Inbound Connection configuration area, specify the JNDI name, which JNDI data source for an existing activation specification in WAS server or you can create it later. For example, `com/eis/AdapterAS`.
5. Click **Finish**.

Configuring a module for Advanced event processing - inbound

To configure a module to use the adapter for Advanced event processing, you use the J2C Bean wizard in Rational Application Developer for WebSphere Software to find an IDoc or set of IDocs, configure the business objects that are generated, and create a deployable module. To use the Advanced event processing interface, you must first add the adapter-supplied transport files to the SAP server.

Selecting business objects and services for Advanced event (inbound) processing

To specify which function you want to process, you provide information in the J2C Bean wizard.

Before you begin

Make sure you have set the connection properties for the J2C Bean wizard.

About this task

Specify search criteria that the J2C Bean wizard uses to discover functions on the SAP server. The J2C Bean wizard returns a list of functions that meet the search criteria.

To specify the search criteria and select one or more functions, use the following procedure.

Procedure

1. In the Object Discovery and Selection window, indicate which IDoc you want to work with.
   a. Expand **AEP**.
   b. Click **Discover IDoc From System** to enable the filter button.
   c. Click the filter button.

   Note: Instead of using the filter button, you can expand **Discover IDoc From System** and select the IDoc from the list. Then skip ahead to step 4 on page 124.

2. From the Filter Properties window, specify information about the IDoc or IDocs:
   a. Select **Discover objects by name** or **Discover objects by description** from the **Object attribute to use for discovery** list.
   b. Type a search string representing the IDoc you want to call.
   c. Select **Basic IDocs** or **Extension IDocs** from the **IDoc type to use for discovery** field.
   d. Indicate the number of functions you want returned by changing the value in the **Maximum number of objects to retrieve** field or by accepting the default value.
   e. Click **OK**.
3. Select the IDoc or IDocs.
   a. Expand **Discover IDoc From System (filtered)**.
   b. Click the IDoc you want to use. If you are working with multiple IDocs, click the names of all the IDocs.
4. Click the arrow button to add the IDoc or IDocs to the **Selected objects** list.
5. In the Configuration Parameters window, perform the following tasks to add the IDoc to the list of business objects to be imported.
   a. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   b. In the **IDoc release version** field, specify the SAP release number to identify the IDoc type you want the J2C Bean wizard to use for creating business objects.
   c. Expand the IDoc name and select one or more nodes to be used as the primary key, or leave the default values selected.
   d. Click **OK**.
6. Click **Next**.

Results

The J2C Bean wizard has returned a list of the function or functions that match the search criteria, and you have selected the function or functions you want to work with.

What to do next

From the Configure Composite Properties window, associate an operation with the IDoc and specify the ABAP function module for the selected operation.

Configuring the selected objects

To configure the business object, you specify information about the object (such as the operation associated with the object).

Before you begin

Make sure you have selected and imported the IDoc.

About this task

To configure the business object, use the following procedure.

Procedure

1. In the Configure Composite Properties window, click an IDoc from the **IDoc to configure** list.

   If you are configuring only one IDoc, this step is not necessary.
2. Click **Add** in the Service operations for selected IDoc section of the window.
3. Select an operation (for example, **Create**), and click **OK**.
4. In the **ABAP function module name for selected operation** field, type the name of the ABAP function module to associate with this operation.
5. If you are working with multiple IDocs, repeat the previous four steps for each IDoc.
6. In the **Business object namespace** field, use the default namespace (http://www.ibm.com/xmlns/prod/websphere/j2ca/sap) except in the following circumstance. If you are adding the business object to an existing module and the module already includes that business object (from an earlier run of the J2C Bean wizard), change the namespace value.

   For example, you could change the namespace to http://www.ibm.com/xmlns/prod/websphere/j2ca/sap1.
7. To indicate where the business object information should be stored, type the path to the location in the **Folder** field. This is an optional step.
8. Click **Finish**.

Results

You have associated an operation with each IDoc and associated an ABAP function module with the object. The Service Generation and Deployment Configuration window is displayed.

What to do next

Generate a deployable module that includes the adapter and the business object.

Setting deployment properties and generating artifacts

After you select and configure business objects for your module, use the J2C Bean wizard to configure properties that the adapter uses to connect to a specific database. The wizard creates a new Java project where all the artifacts and property values are saved.

About this task

This task is performed through the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

Procedure

1. In the **EJB Project name** field, select or create a new EJB project.
   - To select an existing project, click **Browse**. If the desired project name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new project:
     a. Click **New**.
     b. In the EJB Project window, type a project name. For example, MyAdapterInboundEJB.
     c. In the EAR Membership area, click **New** to create a new ear project.

d. In EAR Application Project window, type an EAR project name. For example, MyAdapterInboundEJBEAR
e. Click **Finish** to return to the EJB Project window.
f. Click **Finish**.

2. In the **Package name** field, select or create a new package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the desired package name appears in the **Project Selection** list, select its name.
   - Otherwise, create a new package:
     a. Click **New**.
     b. In the New Java Package window, type a name for the package. For example, myadapterinboundejbpkg.
     c. Click **Finish**.
3. In the **Stateless Session EJB's local business interface name** field, specify the interface name you want to use for your business object. For example, MyAdapterInboundInterface. The interface name is suffixed with "MDB" and it is displayed automatically in the **Message Driven EJB name** field. For example, MyAdapterInboundInterfaceMDB. Similarly, the interface name is suffixed with "SB" and it is displayed automatically in the **Stateless Session EJB name** field. For example, MyAdapterInboundInterfaceSB.
4. In the Inbound Connection configuration area, specify the JNDI name, which JNDI data source for an existing activation specification in WAS server or you can create it later. For example, `com/eis/AdapterAS`.
5. Click **Finish**.

Chapter 5. Configuring the application on WebSphere Application Server

When you are running the adapter in a stand-alone deployment, use the administrative console of the server to start, stop, monitor, and troubleshoot the adapter module. In an application that uses an embedded adapter, the adapter module starts or stops when the application is started or stopped.

Changing configuration properties for embedded adapters

To change configuration properties after you deploy the adapter as part of a module, you use the administrative console of the runtime environment. You can update resource adapter properties (used for general adapter operation), managed connection factory properties (used for outbound processing), and activation specification properties (used for inbound processing).

Setting resource adapter properties for embedded adapters

To set resource adapter properties for your adapter after it has been deployed as part of a module, use the administrative console. You select the name of the property you want to configure and then change or set the value.

Before you begin

Your adapter module must be deployed on WebSphere Application Server.

About this task

Custom properties are default configuration properties shared by all WebSphere adapters.

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Under **Applications>Application Types**, select **WebSphere enterprise applications**.
3. From the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change. The **Configuration** page is displayed.
4. Under **Modules**, click **Manage Modules**.
5. Click **IBM WebSphere Adapter for SAP Software**.
6. From the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **Custom properties**.
8. For each property you want to change, perform the following steps.

   Note: See "Resource adapter properties" on page 174 for more information about these properties.

   a. Click the name of the property. The **Configuration** page for the selected property is displayed.
   b. Change the contents of the **Value** field or type a value, if the field is empty.

You can change the number in the **Value** field and add a description of the property.

c. Click **OK**.

9. Click the **Save** link in the **Messages** box at the top of the window.

Results

The resource adapter properties associated with your adapter module are changed.

Setting managed (J2C) connection factory properties for embedded adapters

To set managed connection factory properties for your adapter after it has been deployed as part of a module, use the administrative console. You select the name of the property you want to configure and then change or set the value.

Before you begin

Your adapter module must be deployed on WebSphere Application Server.

About this task

You use managed connection factory properties to configure the target SAP server instance.

Note: In the administrative console, the properties are referred to as ″J2C connection factory properties.″

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Under **Applications>Application Types**, select **WebSphere enterprise applications**.
3. In the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change.
4. Under **Modules**, click **Manage Modules**.
5. Click **IBM WebSphere Adapter for SAP Software**.
6. In the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **J2C connection factories**.
8. Click the name of the connection factory associated with your adapter module.
9. In the **Additional Properties** list, click **Custom properties**.

   Custom properties are those J2C connection factory properties that are unique to Adapter for SAP Software. Connection pool and advanced connection factory properties are properties you configure if you are developing your own adapter.
10. For each property you want to change, perform the following steps.

    Note: See “Managed connection factory properties” on page 176 for more information about these properties.

a. Click the name of the property.
b. Change the contents of the **Value** field or type a value, if the field is empty.
c. Click **OK**.

11. Click the **Save** link in the **Messages** box at the top of the window.

Results

The managed connection factory properties associated with your adapter module are changed.

Setting activation specification properties for embedded adapters

To set activation specification properties for your adapter after it has been deployed as part of a module, use the administrative console. You select the name of the message endpoint property you want to configure, and then change or set the value.

Before you begin

Your adapter module must be deployed on WebSphere Application Server.

About this task

You use activation specification properties to configure the endpoint for inbound processing.

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Under **Applications>Application Types**, select **WebSphere enterprise applications**.
3. From the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change.
4. Under **Modules**, click **Manage Modules**.
5. Click **IBM WebSphere Adapter for SAP Software**.
6. From the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **J2C activation specifications**.
8. Click the name of the activation specification associated with the adapter module.
9. From the **Additional Properties** list, click **J2C activation specification custom properties**.
10. For each property you want to change, perform the following steps.
    a. Click the name of the property.
    b. Change the contents of the **Value** field or type a value, if the field is empty.
    c. Click **OK**.
11. Click the **Save** link in the **Messages** box at the top of the window.

Results

The activation specification properties associated with your adapter module are changed.

Changing configuration properties for stand-alone adapters

To set configuration properties after you install a stand-alone adapter, you use the administrative console of the runtime environment. You provide general information about the adapter and then set resource adapter properties (which are used for general adapter operation). If the adapter will be used for outbound operations, you create a connection factory and then set properties for it. If the adapter will be used for inbound operations, you create an activation specification and then set properties for it.

Setting resource adapter properties for stand-alone adapters

To set resource adapter properties for your stand-alone adapter after it has been installed on WebSphere Application Server, use the administrative console. You select the name of the property you want to configure and then change or set the value.

Before you begin

Your adapter must be installed on WebSphere Application Server.

About this task

Custom properties are default configuration properties shared by all WebSphere adapters.

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. In the Resource adapters page, click **IBM WebSphere Adapter for SAP Software**.
4. In the **Additional Properties** list, click **Custom properties**.
5. For each property you want to change, perform the following steps.
   a. Click the name of the property.
   b. Change the contents of the **Value** field or type a value, if the field is empty. You can change the number in the **Value** field and add a description of the property.
   c. Click **OK**.
6. Click **Save** in the **Messages** box at the top of the page.

Results

The resource adapter properties associated with your adapter are changed.

Setting managed (J2C) connection factory properties for stand-alone adapters

To set managed connection factory properties for your stand-alone adapter after it has been installed on WebSphere Application Server, use the administrative console. You select the name of the property you want to configure and then change or set the value.

Before you begin

Your adapter must be installed on WebSphere Application Server.

About this task

You use managed connection factory properties to configure the target SAP server instance.

Note: In the administrative console, the properties are referred to as ″J2C connection factory properties.″

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. In the Resource adapters page, click **IBM WebSphere Adapter for SAP Software**.
4. In the **Additional Properties** list, click **J2C connection factories**.
5. If you are going to use an existing connection factory, skip ahead to select from the list of existing connection factories.

   Note: If you selected **Use predefined connection properties** when you used the J2C Bean wizard to configure the adapter module, you do not need to create a connection factory.

   If you are creating a connection factory, perform the following steps:

   a. Click **New**.
   b. In the **General Properties** section of the **Configuration** tab, type a name for the connection factory. For example, you could type `AdapterCF`.
   c. Type a value for **JNDI name**. For example, you could type `com/eis/AdapterCF`.
   d. Optional: Select an authentication alias from the **Component-managed authentication alias** list.
   e. Click **OK**.
   f. Click **Save** in the **Messages** box at the top of the page.

      The newly created connection factory is displayed.
6. In the list of connection factories, click the one you want to use.
7. In the **Additional Properties** list, click **Custom properties**.

   Custom properties are those J2C connection factory properties that are unique to Adapter for SAP Software. Connection pool and advanced connection factory properties are properties you configure if you are developing your own adapter.

8. For each property you want to change, perform the following steps.

   Note: See "Managed connection factory properties" on page 176 for more information about these properties.

   a. Click the name of the property.
   b. Change the contents of the **Value** field or type a value, if the field is empty.
   c. Click **OK**.
9. After you have finished setting properties, click **Apply**.
10. Click **Save** in the **Messages** box at the top of the window.

Results

The managed connection factory properties associated with your adapter are set.

Setting activation specification properties for stand-alone adapters

To set activation specification properties for your stand-alone adapter after it has been installed on WebSphere Application Server, use the administrative console. You select the name of the message endpoint property you want to configure, and then change or set the value.

Before you begin

Your adapter must be installed on WebSphere Application Server.

About this task

You use activation specification properties to configure the endpoint for inbound processing.

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. In the Resource adapters page, click **IBM WebSphere Adapter for SAP Software**.
4. In the **Additional Properties** list, click **J2C activation specifications**.
5. If you are going to use an existing activation specification, skip ahead to select from an existing list of activation specifications.

   Note: If you selected **Use predefined connection properties** when you used the J2C Bean wizard to configure the adapter module, you do not need to create an activation specification.

   If you are creating an activation specification, perform the following steps:

   a. Click **New**.
   b. In the **General Properties** section of the **Configuration** tab, type a name for the activation specification. For example, you could type `AdapterAS`.
   c. Type a value for **JNDI name**. For example, you could type `com/eis/AdapterAS`.

d. Optional: Select an authentication alias from the **Authentication alias** list.
e. Select a message listener type. The available listener types correspond to:
   - The ALE inbound processing interface
   - The ALE inbound processing interface with local transaction support
   - The BAPI inbound processing interface
   - The Advanced event processing inbound interface
f. Click **OK**.
g. Click **Save** in the **Messages** box at the top of the page.
   The newly created activation specification is displayed.

6. In the list of activation specifications, click the one you want to use.
7. In the Additional Properties list, click **J2C activation specification custom properties**.
8. For each property you want to set, perform the following steps.
   a. Click the name of the property.
   b. Change the contents of the **Value** field or type a value, if the field is empty.
   c. Click **OK**.
9. After you have finished setting properties, click **Apply**.
10. Click **Save** in the **Messages** box at the top of the page.

Results

The activation specification properties associated with your adapter are set.

Managing Advanced event processing

To manage the Advanced event processing interface, use the IBM WebSphere BI Station tool. You can view and maintain events in the current events queue, future events queue, and archive events queue, and you can view and maintain adapter log files. In addition, you can maintain the SAP gateway service connections.

Displaying the current events queue

You can display the outgoing current events queue to check for events that have not yet been retrieved by WebSphere Adapter for SAP Software.

Before you begin

Make sure you have successfully installed the IBM WebSphere BI Station tool on the SAP server.

About this task

The events in the current events queue are awaiting retrieval by the adapter. You can display the queue to check the status of the events.

To display the contents of the current events queue, use the following procedure.

Procedure

1. If IBM WebSphere BI Station is not currently displayed, enter transaction `/n/CWLD/HOME_AEP`.
2. To display the Management page, click **Management**.

3. Under **Event Queues**, click **Current Events**.
4. Display the current events queue by performing one of the following steps from the Current Event Selection page:
   - To display all events in the current events queue, click **Execute**.
   - To limit the number of events that are displayed, enter values in one or more fields, or use the arrow keys to select values for the fields, and click **Execute**.

     For example, to display only those entries associated with a particular business object, enter the name of the business object in the **Object Name** field, or click the **Object Name** field and select a value from the list.

Results

A list of events is displayed.

Displaying the future events queue

You can display the future events queue to check for events that have not yet been transferred to the current events queue.

Before you begin

Make sure you have successfully installed the IBM WebSphere BI Station tool on the SAP server.

About this task

The events in the future events queue are awaiting transfer to the current events queue. You can display the queue to check the status of the events.

To display the contents of the future events queue, use the following procedure.

Procedure

1. If IBM WebSphere BI Station is not currently displayed, enter transaction `/n/CWLD/HOME_AEP`.
2. To display the Management page, click **Management**.
3. Under **Event Queues**, click **Future Events**.
4. Display the future events queue by performing one of the following steps from the Future Event Selection page:
   - To display all events in the future events queue, click **Execute**.
   - To limit the number of events that are displayed, enter values in one or more fields, or use the arrow keys to select values for the fields, and click **Execute**.

     For example, to display only those entries associated with a particular business object, enter the name of the business object in the **Object Name** field, or click the **Object Name** field and select a value from the list.

Results

A list of events is displayed.

Maintaining the archive table

Using the IBM WebSphere BI Station tool, you can display the archive table and determine the status of archived events. From the table, you can identify events that need to be resubmitted for polling when a runtime environment subscribes to them.

Before you begin

Make sure you have successfully installed the IBM WebSphere BI Station tool on the SAP server.

About this task

When you display events in the archive table, you can resubmit the events for processing, or you can delete the events from the table.

To maintain the archive table, perform one or more of the following steps.

Procedure

1. If IBM WebSphere BI Station is not currently displayed, enter transaction `/n/CWLD/HOME_AEP`.
2. To display the Management page, click **Management**.
3. Under **Event Queues**, click **Archived Events**.
4. Display the event queue by performing one of the following steps from the Archived Event Selection page:
   a. To display all events, click the Execute button (F8).
   b. To limit the number of events that are displayed, enter values in one or more fields, or use the arrow keys to select values for the fields.

      For example, to display only those entries associated with a particular business object, enter the name of the business object in the **Object Name** field, or click **Object Name**, click the arrow button (F4), and then select the name from the list.

Results

A list of events is displayed.

What to do next

Resubmit one or more events for processing, or delete one or more events.

Resubmitting archived events

You can resubmit one or more events from the archive table to the event queue for reprocessing.

Before you begin

The Archived Events page should be displayed.

About this task

Resubmitting events moves the events from the archive table to the event table; however, the events do not pass through event distribution, event restriction, or event priority.

To resubmit one or more events, perform the following procedure.

Procedure

1. To select the event to be resubmitted, select the check box next to the name of the event. You can select multiple events.
2. Click **Resubmit**.

Results

The status of the operation is displayed.

Deleting events from the archive table

You can delete one or more events from the archive table. You can delete the files from the Management page, or you can schedule their deletion.

Before you begin

The Management page of IBM WebSphere BI Station should be displayed.

About this task

To delete events from the archive table, perform the following steps:

Procedure

1. If IBM WebSphere BI Station is not currently displayed, enter transaction `/n/CWLD/HOME_AEP`.
2. To display the Management page, click **Management**.
3. Under **Maintenance**, click **Delete Event Archive**.
4. From the WebSphere BI Delete Entries from Event Archive Table page, enter values for one or more fields to limit the events that are deleted.

   For example, to delete only those entries associated with a particular business object, enter the name of the business object in the **Object Name** field, or click **Object Name**, click the arrow button (F4), and then select the name from the list.
5. Click the Execute button (F8).

   Note: To schedule automatic deletion of archive events, contact your Basis administrator and schedule report /CWLD/TRUN_EVENT_ARCHIVE_TAB.

Results

The event or events are deleted.

Managing the adapter log file

The adapter log in the SAP application displays in reverse chronological order all events and errors that relate to the SAP server, such as Create or Update operations, or events that arrive in the event queue. The log file lists the date, time, and event for each log entry. The log file is a good source to start troubleshooting problems.

Setting logging options

You can specify the level of detail you want logged in the adapter log file, as well as the number of entries and type of data you want displayed.

Before you begin

Make sure you have successfully installed the IBM WebSphere BI Station tool on the SAP server.

About this task

To set the logging options, use the following procedure.

Procedure

1. If IBM WebSphere BI Station is not currently displayed, enter transaction `/n/CWLD/HOME_AEP`.
2. Click **Configuration**.
3. To set the logging level, select one of the values under **Logging Level**. The four levels of logging are shown in the following table:

Table 11. Logging levels

Level	Description	Recommended use
0	Off	Not recommended
1	Log only warnings and errors	Production system
2	Log every event with minimal information	
3	Log each event in detail, including every attribute of every business object	Development or debugging system

4. To change how many events are displayed, type a value in the **Number of entries to display in log** field.
5. To display only errors in the log, select **Display errors only**.
6. To display only entries for the user listed next to **User Name**, select **Display entries for this user**.
7. To specify how much (or how little) detail to display in the log, select one of the values under **Default Level of Detail to Display**.

Results

You have set the configuration settings that will be used when the log is displayed.

Displaying the adapter log

To view recently processed objects and the details associated with them, display the adapter log.

Before you begin

Make sure you have successfully installed the IBM WebSphere BI Station tool on the SAP server.

About this task

You can specify how much detail you want displayed, and you can filter the data so that only certain types of information are displayed.

To display the adapter log, use the following procedure.

Procedure

1. If IBM WebSphere BI Station is not currently displayed, enter transaction /n/CWLD/HOME_AEP.
2. To display the Management page, click **Management**.
3. Under **Activity**, click **Log**.
4. To change the amount of information that is displayed, click either **Fewer Details** or **More Details**.
5. To display only specific information, click **Filter Data**, enter values in the fields, and click **Filter**.

   You can choose to display log entries associated with a specific user or with selected objects. You can display entries for a range of dates or a range of numbers. You can indicate how many entries should be displayed and whether to show errors and warnings only.

Results

The log is displayed.

Limiting the size of the adapter log

The adapter log can, over time, take up a significant amount of disk space. To save disk space, you can set this log to automatically truncate. When you set automatic truncation, by default SAP prints the truncated entries to the default printer of the user who set up the job. Therefore, you might also want to control the print options.

Before you begin

Make sure you have successfully installed the IBM WebSphere BI Station tool on the SAP server.

About this task

To limit the size of the adapter log, use the following procedure.

Procedure

1. If IBM WebSphere BI Station is not currently displayed, enter transaction /n/CWLD/HOME_AEP.
2. To display the Management page, click **Management**.
3. Under **Maintenance**, click **Delete Log**.
4. In the WebSphere BI Delete Log Entries page, enter values to indicate which log entries you want to delete.

   You can delete a range of entries or the entries associated with a specific object. You can delete entries associated with a specific user or entries that were logged within a range of dates. You can also indicate that only entries older than a certain number of days should be deleted, and you can specify that a certain number of the most recent entries not be deleted.

   The entries being deleted from the log are saved to the file specified in the **Output truncated data to** field.
5. Click the Execute button.

   Note: To schedule the automatic truncation of the event log, set up the truncation options and contact your Basis administrator to schedule report /CWLD/DELETE_LOG.

Results

The specified log entries are deleted.

Monitoring SAP gateway connections

You can monitor the SAP gateway service connections between the adapter and the SAP application. Each entry displays information such as adapter host name, user name, and connection status.

Before you begin

Make sure you have successfully installed the IBM WebSphere BI Station tool on the SAP server.

About this task

To monitor the gateway connections, use the following procedure.

Procedure

1. If IBM WebSphere BI Station is not currently displayed, enter transaction `/n/CWLD/HOME_AEP`.
2. To display the Management page, click **Management**.
3. Under **Activity**, click **Gateway**.
4. Click a server name to see more details.

Results

A list of active connections is displayed.

Adding dependency libraries to the deployed resource adapter

The deployed resource adapter running in the WebSphere Application Server requires the same dependency libraries as it does in Rational Application Developer for WebSphere Software to process requests. The method for adding these library files depends on the mode of the resource adapter deployment: standalone or embedded in the EAR file.

Standalone deployment

The dependency libraries may be added to the resource adapter deployed standalone either during initial deployment of the RAR file or by configuring the Resource Adapter properties after deployment.

About this task

To set the values during initial deployment of the RAR file, specify Class path and Native path locations. Class path is used to point to JAR files, and Native path is used to point to native libraries, such as *.dll, *.so.

To set the dependency library path files after the adapter has been installed on WebSphere Application Server, use the administrative console to modify the values for the Resource Adapter.

EAR deployment

For the rare case when the connector needs to be embedded in the EAR file, the dependant libraries are added as shared libraries. Define the appropriate shared library containing external dependencies and associate them with the EAR file.

About this task

There are two methods to do this task:

- Using enhanced EAR editor in Rational Application Developer for WebSphere Software
- Using administrative console of the WebSphere Application Server

Using enhanced EAR editor

You can use the EAR editor in Rational Application Developer for WebSphere Software to add the dependency libraries.

About this task

To create shared libraries using the EAR editor, use the following procedure.

Procedure

1. Open Enhanced EAR editor.
2. Click **Deployment** tab.
3. Expand **Shared Library** Shared Library section.
4. Click **Add** to add new shared library.
5. Specify Shared library parameters and click **OK**.
6. Deploy the EAR to the server.

Using administrative console of the WebSphere Application Server

You can use the administrative console of the WebSphere Application Server to add the dependency libraries.

Before you begin

Make sure dependent files are available on the server machine in the separate folder. If needed, copy dependent files on the server machine.

Procedure

1. Define WebSphere variables to point to appropriate folders.
2. Define the shared library through the server administrative console; you can specify it using WebSphere variables defined in above step 1.
3. Deploy the EAR to the server.
4. Configure the EAR to reference defined shared library.

Chapter 6. Troubleshooting and support

Common troubleshooting techniques and self-help information help you identify and solve problems quickly.

Support for the Log and Trace Analyzer

The adapter creates log and trace files that can be viewed with the Log and Trace Analyzer.

The Log and Trace Analyzer can filter log and trace files to isolate the messages and trace information for the adapter. It can also highlight the adapter's messages and trace information in the log viewer.

The adapter's component ID for filtering and highlighting is a string composed of the characters `SAPRA` plus the value of the adapter ID property. For example, if the adapter ID property is set to `001`, the component ID is `SAPRA001`.

If you run multiple instances of the same adapter, make sure that the first eight characters of the adapter ID property are unique for each instance so that you can correlate log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate log and trace information to a particular instance of an adapter. To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for SAP Software to `001` and `002`. The component IDs for those instances, `SAPRA001` and `SAPRA002`, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. For example, suppose you set the adapter ID properties of two instances to `Instance01` and `Instance02`. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to `SAPRAInstance`.

For outbound processing, the adapter ID property is located in both the resource adapter and managed connection factory property groups. If you update the adapter ID property after using the J2C Bean wizard to configure the adapter for outbound processing, be sure to set the resource adapter and managed connection factory properties consistently, to prevent inconsistent marking of the log and trace entries. For inbound processing, the adapter ID property is located only in the resource adapter properties, so this consideration does not apply.

For more information about the adapter ID property, see "Adapter ID (AdapterID)" on page 175.

Configuring logging and tracing

Configure logging and tracing to suit your requirements. Enable logging for the adapter to control the status of event processing. Change the adapter log and trace file names to separate them from other log and trace files.

About this task

Configuring logging properties

Use the administrative console to enable logging and to set the output properties for a log, including the location, level of detail, and output format of the log.

About this task

Before the adapters can log monitored events, you must specify the service component event points that you want to monitor, what level of detail you require for each event, and format of the output used to publish the events to the logs. Use the administrative console to perform the following tasks:

- Enable or disable a particular event log
- Specify the level of detail in a log
- Specify where log files are stored and how many log files are kept
- Specify the format for log output

  If you set the output for log analyzer format, you can open trace output using the Log Analyzer tool, which is an application included with your application server. This is useful if you are trying to correlate traces from two different server processes, because it allows you to use the merge capability of the Log Analyzer.

For more information about monitoring on an application server, including service components and event points, see the documentation for your application server.

You can change the log configuration statically or dynamically. Static configuration takes effect when you start or restart the application server. Dynamic, or runtime, configuration changes apply immediately.

When a log is created, the detail level for that log is set from the configuration data. If no configuration data is available for a particular log name, the level for that log is obtained from the parent of the log. If no configuration data exists for the parent log, the parent of that log is checked, and so on up the tree, until a log with a non-null level value is found. When you change the level of a log, the change is propagated to the children of the log, which recursively propagate the change to their children, as necessary.

To enable logging and set the output properties for a log, use the following procedure.

Procedure

1. In the navigation pane of the administrative console, click **Servers** → **WebSphere application servers**.
2. Click the name of the server that you want to work with.
3. Under **Troubleshooting**, click **Logging and tracing**.
4. Click **Change Log Detail Levels**.
5. Specify when you want the change to take effect:
   - For a static change to the configuration, click the **Configuration** tab.
   - For a dynamic change to the configuration, click the **Runtime** tab.
6. Click the names of the packages whose logging level you want to modify. The package names for WebSphere Adapters start with **com.ibm.j2ca.***:
   - For the adapter base component, select **com.ibm.j2ca.base.***.
   - For the adapter base component and all deployed adapters, select **com.ibm.j2ca.***.

- For the Adapter for SAP Software only, select the **com.ibm.j2ca.sap.*** package.

7. Select the logging level.

Logging Level	Description
Fatal	The task cannot continue or the component cannot function.
Severe	The task cannot continue, but the component can still function. This logging level also includes conditions that indicate an impending fatal error, that is, situations that strongly suggest that resources are on the verge of being depleted.
Warning	A potential error has occurred or a severe error is impending. This logging level also includes conditions that indicate a progressive failure, for example, the potential leaking of resources.
Audit	A significant event has occurred that affects the server state or resources.
Info	The task is running. This logging level includes general information outlining the overall progress of a task.
Config	The status of a configuration is reported or a configuration change has occurred.
Detail	The subtask is running. This logging level includes general information detailing the progress of a subtask.

8. Click **Apply**.
9. Click **OK**.
10. To have static configuration changes take effect, stop and then restart the application server.

Results

Log entries from this point forward contain the specified level of information for the selected adapter components.

Changing the log and trace file names

To keep the adapter log and trace information separate from other processes, use the administrative console to change the file names. By default, log and trace information for all processes and applications on a application server is written to the SystemOut.log and trace.log files, respectively.

Before you begin

You can change the log and trace file names at any time after the adapter module has been deployed to an application server.

About this task

You can change the log and trace file names statically or dynamically. Static changes take effect when you start or restart the application server. Dynamic or run time changes apply immediately.

Log and trace files are in the *install_root*/profiles/*profile_name*/logs/*server_name* folder.

To set or change the log and trace file names, use the following procedure.

Procedure

1. In the navigation pane of the administrative console, select **Applications > Application Types > WebSphere application servers**.
2. In the Enterprise Applications list, click the name of the adapter application. This is the name of the EAR file for the adapter, but without the .ear file extension. For example, if the EAR file is named Accounting_OutboundApp.ear, then click **Accounting_OutboundApp**.
3. In the Configuration tab, in the Modules list, click **Manage Modules**.
4. In the list of modules, click IBM WebSphere Adapter for SAP Software.
5. In the Configuration tab, under Additional Properties, click **Resource Adapter**.
6. In the Configuration tab, under Additional Properties, click **Custom properties**.
7. In the Custom Properties table, change the file names.
   a. Click either **logFilename** to change the name of the log file or **traceFilename** to change the name of the trace file.
   b. In the Configuration tab, type the new name in the **Value** field. By default, the log file is called SystemOut.log and the trace file is called trace.log.
   c. Click **Apply** or **OK**. Your changes are saved on your local machine.
   d. To save your changes to the master configuration on the server, use one of the following procedures:
      - **Static change**: Stop and restart the server. This method allows you to make changes, but those changes do not take effect until you stop and start the server.
      - **Dynamic change**: Click the **Save** link in the Messages box above the Custom properties table. Click **Save** again when prompted. This method allows you to make changes that take effect right away.

Detecting errors during outbound processing

To detect errors such as invalid data or invalid state that occur during outbound processing, you set up business-object application-specific data.

Before you begin

Make sure you have determined which errors you want to detect.

About this task

During outbound processing, the adapter can automatically detect errors generated by the SAP JCo interface. To detect other types of errors returned by the RFC interface (for example, to be able to validate the data that is returned) you must define values for application-specific data (metadata) at the business-object level.

To set up the business-object level metadata to detect errors, use the following procedure.

Procedure

1. Identify the parameters that define RFC error codes and their possible values.
2. Display the business object in the XML Schema Editor.
3. From the Properties tab, in the Extensions section, select **sapBAPIBusinessObjectTypeMetadata**.

4. Click **Add**, and select **sapasi:ErrorConfiguration**.
5. Add the application-specific information for ErrorParameter, ErrorCode, and ErrorDetail to the business object by right-clicking **sapasi:ErrorConfiguration**, clicking **New**, and selecting **sapasi:ErrorParameter**, **sapasi:ErrorCode**, and **sapasi:ErrorDetail**.
   - ErrorParameter is the XPATH to the property that returns the error codes.
   - ErrorCode contains all possible values (for example, `E`, `ERROR`, and `NODATA`) returned in the property referred to by ErrorParameter.
   - ErrorDetail is the XPATH to the property that contains details about the error.

   If the values defined in the ErrorCode property match the error parameter values after RFC executes the call, an error message with detailed information is generated. The detail is derived from the ErrorDetail property.

   Error handling application-specific information must be manually maintained.

Results

Your top-level business object contains properties that enable it to detect RFC errors.

Resolving errors during Query interface for SAP Software processing

If the exception `Error in ASSIGN statement in the program SAPLSDTX` is generated during Query interface for SAP Software processing, you must change the function used by the adapter to retrieve data from SAP tables.

About this task

On non-Unicode systems, the default function used to retrieve data from SAP tables (RFC_READ_TABLE) might produce an exception. To avoid the problem, you can create another function on the SAP server and then indicate, during adapter configuration, that the adapter should use this newly created function to retrieve data.

To create the custom retrieve function and specify it during configuration, perform the following tasks:

Procedure

1. Follow the steps listed in SAP note 758278 to make a copy of RFC_READ_TABLE and modify the copy per the note.
2. Configure a module for Query interface for SAP Software in the J2C Bean wizard. On the Configure Composite Properties window, click the **Advanced** button and provide the name of the custom function you created in step 1.

Resolving memory-related issues

You can increase the WebSphere Application Server memory limit if you encounter memory-related issues.

Increase the memory limit if you encounter the following problems:

- You see an out-of-memory error when a very large IDoc is sent from the SAP server to WebSphere Application Server.
- You see the error message `JCO Server could not unmarshall tables`.

To increase the memory limit, use the Jvm arguments for the initial (ms) and maximum (mx) size (for example, `-mx512m -mx256m`) in the server startup command.

First-failure data capture (FFDC) support

The adapter supports first-failure data capture (FFDC), which provides persistent records of failures and significant software incidents that occur during run time in WebSphere Application Server.

The FFDC feature runs in the background and collects events and errors that occur at run time. The feature provides a means for associating failures to one another, allowing software to link the effects of a failure to their causes, and thereby facilitate the quick location of the root cause of a failure. The data that is captured can be used to identify exception processing that occurred during the adapter run time.

When a problem occurs, the adapter writes exception messages and context data to a log file, which is located in the *install_root*/profiles/*profile*/logs/ffdc directory.

For more information about first-failure data capture (FFDC), see the WebSphere Application Server documentation.

XAResourceNotAvailableException

When the application server log contains repeated reports of the com.ibm.ws.Transaction.XAResourceNotAvailableException exception, remove transaction logs to correct the problem.

Symptom:

When the adapter starts, the following exception is repeatedly logged in the application server log file:

com.ibm.ws.Transaction.XAResourceNotAvailableException

Problem:

A resource was removed while the application server was committing or rolling back a transaction for that resource. When the adapter starts, it tries to recover the transaction but cannot because the resource was removed.

Solution:

To correct this problem, use the following procedure:

1. Stop the application server.
2. Delete the transaction log file that contains the transaction. Use the information in the exception trace to identify the transaction. This prevents the server from trying to recover those transactions.

   Note: In a test or development environment, you can generally delete all of the transaction logs. In Rational Application Developer for WebSphere Software, delete the files and subdirectories of the transaction log directory, *server_install_directory*\profiles*profile_name*\tranlog.

In a production environment, delete only the transactions that represent events that you do not need to process. One way to do this is to reinstall the adapter, pointing it to the original event database used, and deleting only the transactions you do not need. Another approach is to delete the transactions from either the log1 or log2 file in the following directory:

server_install_directory\profiles*profile_name*\tranlog*node_name**server_name*\transaction\tranlog

3. Start the application server.

Self-help resources

Use the resources of IBM software support to get the most current support information, obtain technical documentation, download support tools and fixes, and avoid problems with WebSphere Adapters. The self-help resources also help you diagnose problems with the adapter and provide information about how to contact IBM software support.

Support Web site

The WebSphere Adapters software support Web site at http://www.ibm.com/software/integration/wbiadapters/support/ provides links to many resources to help you learn about, use, and troubleshoot WebSphere Adapters, including the following types of

- Flashes (alerts about the product)
- Technical information including the product information center, manuals, IBM Redbooks®, and whitepapers
- Educational offerings
- Technotes

Recommended fixes

A list of recommended fixes you should apply is available at the following location: http://www.ibm.com/support/docview.wss?fdoc=aimadp&rs=695&uid=swg27010397

Technotes

Technotes provide the most current documentation about the Adapter for SAP Software, including the following topics:

- Problems and their currently available solutions
- Answers to frequently asked questions
- How-to information about installing, configuring, using, and troubleshooting the adapter
- *IBM Software Support Handbook*

For a list of technotes for WebSphere Adapters, visit this address:

http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8&dc=DB520+D800+D900+DA900+DA800+DB560&dtm

Plug-in for IBM Support Assistant

Adapter for SAP Software provides a plug-in for IBM Support Assistant, which is a free, local software serviceability workbench. The plug-in supports the dynamic trace feature. For information about installing or using IBM Support Assistant, visit this address:

http://www.ibm.com/software/support/isa/

Chapter 7. Reference information

To support you in your tasks, reference information includes details about business objects that are generated by the J2C Bean wizard and information about adapter properties, including those that support bidirectional transformation. It also includes pointers to adapter messages and related product information.

Business object information

A business object contains application-specific information (metadata) about how the adapter should process the business object as well as the operation to be performed on the business object. The name of the business object is generated by the J2C Bean wizard in accordance with the naming convention for the adapter.

Application-specific information

Application-specific information (ASI) is metadata that specifies adapter-dependent information about how to process business objects for the adapter for SAP Software. When the J2C Bean wizard generates a business object, it automatically generates a business object definition, which is saved as an XSD (XML Schema Definition) file. The business object definition contains the application-specific information for that business object.

BAPI business object application-specific information

BAPI application-specific information (ASI) is metadata that specifies adapter-dependent information about how to process BAPI business objects for the WebSphere Adapter for SAP Software.

Business object-level metadata for BAPI

WebSphere Adapter for SAP Software uses application-specific information (ASI) to create queries for Create, Retrieve, Update, and Delete operations. ASI for BAPI is generated by the J2C Bean wizard at the following levels: the business-object level, the operation-level and the property-level.

The sections that follow describe the metadata elements for each level.

Business object-level metadata defines the top-level wrapper of the business object.

The following table lists and describes the business object-level metadata elements for a BAPI business object.

Table 12. Metadata elements: Wrapper of a BAPI business object

Metadata element	Description
Type	The business object type. For a simple BAPI, the value is `BAPI`. For a BAPI work unit business object, this value is `BAPITXN`. For a BAPI result set, this value is `BAPIRS`.

Table 12. Metadata elements: Wrapper of a BAPI business object (continued)

Metadata element	Description
Operation	The valid operations include Create, Update, Delete, and Retrieve. The specified operation metadata is defined in the sapBAPIOperationTypeMetadata tag and contains the following: • `MethodName`: Name of the BAPI associated with the operation. • `Name`: Name of the operation.

Property-level metadata for BAPI business objects

Property-level metadata represents child objects or an array of child objects.

The following table describes the metadata elements of a complex property (child) or structure or table property (an array of child objects).

Table 13. Property-level metadata elements: BAPI business object

Metadata element	Description
FieldName	The BAPI field name as represented in SAP.
FieldType	The type of the property as it exists in SAP.
PrimaryKey	An indication about whether this property is a primary key.
ParameterType	The direction of the mapping. • If the value is `IN`, the property is mapped from the business object to the BAPI. • If the value is `OUT`, the property is mapped from the BAPI in the SAP system to the business object. • If the value is `INOUT`, the property is mapped both ways (BAPI to business object and business object to BAPI).
MaxLength	The length of the field.
ForeignKey	The foreign-key relationship. This element applies only to BAPI result sets.
DecimalPlaces	For fields with a FieldType of `Decimal`, the value of the precision level. This value is extracted from metadata on the SAP server.
Description	The description of the field. This value is extracted from metadata on the SAP server.

Operation-level metadata for BAPI business objects

Operation-level metadata specifies the method name of the BAPI in the SAP system. This name is used by the adapter to determine the action to take on the BAPI.

The following table describes the operation-level metadata elements of a BAPI business object.

Table 14. Operation-level metadata elements: BAPI business object

Metadata element	Description
MethodName	The name of the BAPI call (method) in the SAP system.
Name	The name of the business object operation associated with the MethodName.

ALE business object application-specific information

ALE application-specific information (ASI) is metadata that specifies adapter-dependent information about how to process ALE business objects for the adapter for SAP Software.

The type of metadata that is generated depends on whether you are using the ALE interface or the ALE pass-through IDoc interface:

- **ALE interface**

  The WebSphere Adapter for SAP Software uses application-specific information (ASI) to create queries for Create, Retrieve, Update, and Delete operations.

  ASI for objects generated with the ALE interface is available at the following levels:

  – The IDoc business-object level (for individual IDocs)
  – The IDoc wrapper business-object level (for IDoc packets)
  – The operation level for individual IDoc business objects
  – The property level

  For ALE inbound processing, the adapter for SAP Software uses ASI to determine which of the supported operations (Create, Retrieve, Update, or Delete) to run on the endpoint.

  Note: There is no metadata at the IDoc Data Record or IDoc Control Record child business object-level.

- **ALE pass-through IDoc interface**

  ASI for objects generated with the ALE pass-through IDoc interface is available at the following levels:

  – The IDoc business-object level
  – The property level

The sections that follow describe the metadata elements for each level.

Business-object-level metadata for ALE

- **ALE interface**

  Business object-level metadata for ALE interface business objects defines the top-level wrapper of an IDoc.

  The following table describes the business-object metadata elements of an ALE business object.

Table 15. Business object-level metadata elements: ALE business object

Metadata element	Description
SplitIDocPacket	For inbound operations, an indication of whether the IDoc packet needs to be split into individual IDocs. The possible values are `true` or `false`. If you select the corresponding property (check box) in the J2C Bean wizard, make sure you set this property to `true`.
Type	The business object type. Possible values are `IDOC` or `UNPARSEDIDOC`.

Table 15. Business object-level metadata elements: ALE business object (continued)

Metadata element	Description
Operation	Each *outbound* operation contains the following parameters: **Name** Name of the operation, which for outbound processing is always `Execute`. Each *inbound* operation contains the following parameters: **Name** Name of the operation (`Create`, `Update`, or `Delete`). **MsgType** The message type configured for the IDoc. **MsgCode** The message code configured for the IDoc. **MsgFunction** The message function configured for the IDoc.

- **ALE pass-through IDoc interface**

  Business object-level metadata for ALE pass-through IDoc interface business objects defines the top-level wrapper of an IDoc.

  The following table describes the business-object metadata elements of an ALE pass-through IDoc interface business object.

Table 16. Business object-level metadata elements: Generic IDoc business object

Metadata element	Description
SplitIDocPacket	For inbound operations, an indication of whether the IDoc packet needs to be split into individual IDocs. The possible values are `true` or `false`. If you select the corresponding property (check box) in the J2C Bean wizard, make sure you set this property to `true`.
Type	The business object type. For a generic IDoc, this value is `PASSTHROUGHIDOC`.

Property-level metadata for ALE business objects

Property-level metadata either represents child objects or an array of child objects.

The following table describes the property-level metadata elements of an ALE business object or an ALE pass-through IDoc interface business object.

Table 17. Property-level metadata elements: ALE business object

Metadata element	Description
FieldName	The actual IDoc field name in SAP.
SegmentHierarchy	The hierarchy of the segment in the IDoc.
Offset	The offset value of the current property in the IDoc.
PrimaryKey	An indication of whether this property is a primary key.
ForeignBOKeyRef	The xpath to the primary key on the control or data record business object property, which you set using the J2C Bean wizard.
MaxLength	The length of the field.

Operation-level metadata for ALE business objects

Operation-level metadata for an ALE business object specifies the operation that posts the IDoc object to the SAP application.

The following table describes the operation-level metadata elements of an ALE business object.

Note: Outbound objects use only the Name metadata element. The MsgType, MsgCode, and MsgFunction elements are used for inbound objects only.

Table 18. Operation-level metadata elements: ALE business object

Metadata element	Description
Name	The name of the operation.
MsgType	The message type configured for the IDoc (for inbound objects only).
MsgCode	The message code configured for the IDoc (for inbound objects only).
MsgFunction	The message function configured for the IDoc (for inbound objects only).

Query interface for SAP Software business objects application-specific information

Query interface for SAP Software application-specific information (ASI) is metadata that specifies adapter-dependent information about how to process Query interface for SAP Software business objects for WebSphere Adapter for SAP Software.

Business object-level metadata for Query interface for SAP Software

The adapter for SAP Software uses application-specific information (ASI) to create queries for Create, Retrieve, Update, and Delete operations. ASI for Query interface for SAP Software is generated by the J2C Bean wizard at the following levels: the table and query business-object level and at the property-level.

The sections that follow describe the metadata elements for each level.

The following table describes the business object-level metadata elements of a Query interface for SAP Software table business object.

Table 19. Business object-level metadata elements: Query interface for SAP Software table business object

Metadata element	Description
TableName	The name of the table that this business object represents.
Type	The interface type the business object is supporting, which for the Query interface for SAP Software is `QISS`.

Property-level metadata for Query interface for SAP Software business objects

Property-level metadata represents child objects or an array of child objects.

The following table describes the property-level metadata elements of a Query interface business object.

Table 20. Property-level metadata elements: Query interface for SAP Software business object

Metadata element	Description
ColumnName	The name of the business-object parameter, which is the actual column name in the SAP table.
PrimaryKey	An indication of whether this property is a primary key.
ForeignKey	The foreign key relationship (if this property is a key), which is the reference to the parent table key parameter.
MaxLength	The length of the field.

Advanced event processing business object application-specific information

Advanced event processing application-specific information (ASI) is metadata that specifies adapter-dependent information about how to process business objects for the adapter for SAP Software.

The adapter for SAP Software uses application-specific information (ASI) to create queries for Create, Retrieve, Update, and Delete operations. ASI for Advanced event processing business objects is generated by the J2C Bean wizard at the following levels: the IDoc business-object level (for individual IDocs), the Operation-level for individual IDoc business objects, and the property level.

Note: There is no metadata at the IDoc Data Record or IDoc Control Record child business object-level.

The sections that follow describe the metadata elements for each level.

Business-object-level metadata for Advanced event processing

Business object-level metadata for Advanced event processing business objects defines the top-level wrapper of an IDoc.

The following table describes the business object-level metadata elements of an Advanced event processing business object.

Table 21. Business object-level metadata elements: Advanced event processing

Metadata element	Description
Type	The business object type. The business object type will always be `AEP`.

Table 21. Business object-level metadata elements: Advanced event processing (continued)

Metadata element	Description
Operation	Each *outbound* operation contains the following parameters: **Name** Name of the operation (`Create`, `Update`, `Delete` or `Retrieve`) **MethodName** The name of the Advanced event processing handler for the operation. **RouterName** The name of the router. Each *inbound* operation contains the following parameters: **Name** Name of the operation (`Create`, `Update`, or `Delete`). **MethodName** The name of the Advanced event processing handler for the operation. **RouterName** The name of the router.

For AEP inbound processing, **MethodName** should represent a method that retrieves data from the SAP system. The data retrieved might correspond to a Create, Update or Delete operation. For example, when you *create* a customer in the SAP system, this operation generates an event in the AEP event table (with CustomerID as key). The AEP inbound processing retrieves the data for the customer that was created and sends it to endpoint. A similar processing sequence would occur for customer update or customer delete operations in the SAP system.

Property-level metadata for Advanced event processing business objects

Property-level metadata can represent either child objects or an array of child objects.

The following table describes the property-level metadata elements of an Advanced event processing business object.

Table 22. Property-level metadata elements: Advanced event processing business object

Metadata element	Description
IDOCName	Name of the IDOC
FieldName	Actual BAPI Field name as represented in SAP
PrimaryKey	An indication of whether this property is a primary key.
ForeignKey	Foreign key relationship
MaxLength	The length of the field. The adapter retrieves the number of bytes used by the parameter type and not the actual number of characters or digits. This results in a mismatch between the maxlength ASI property value and the actual length in SAP with import and export parameters of a BAPI.

Operation-level metadata for Advanced event processing business objects

Operation-level metadata for an Advanced event processing business object specifies the operation that posts the IDoc object to the SAP application.

The following table describes the application-specific metadata elements of an Advanced event processing business object operation.

Note: Outbound objects use only the Name metadata element.

Table 23. Operation-level metadata elements: Advanced event processing business object

Metadata element	Description
Name	The name of the operation.
MethodName	The name of the ABAP handler for this operation.
RouterName	The name of the router.

Supported data operations

For outbound processing, an operation is the name of the action *implemented by the adapter* so that the client application component can perform the operation on the SAP server. The adapter uses the application-specific information (ASI) inside the business object definition to implement the operation. The name of the operation typically indicates the type of action to be implemented, such as *create* or *update*. For inbound processing, adapters implement an operation by delivering events to their endpoints. For inbound processing, the action associated with the event varies depending on the interface (ALE or Advanced event processing). When ALE is the interface, the action is pushed to the adapter and the adapter delivers the event to an endpoint. When Advanced event processing is the interface, the event status is polled by the adapter and processed accordingly.

Supported data operations on BAPI business objects

The operation of a BAPI business object is the name of the BAPI call that an adapter issues on the SAP server during outbound processing. The BAPI method determines the operation associated with it. The adapter uses the application-specific information (ASI) inside the business object definition to implement the operation.

BAPIs and BAPI work unit

Operations of a business object are invoked by the component that makes calls to SAP through the adapter. The SAP JCo APIs are used to make the call to the SAP system.

The following table defines operations that the adapter supports for BAPIs and BAPI work unit.

Note: The definitions listed in the table are the *expected* uses for the operations. The action taken in the SAP application is based on the meaning of the BAPI itself.

Table 24. Supported operations: BAPI business objects

Operation	Definition
Create	The top-level business object and all contained children are created.

Table 24. Supported operations: BAPI business objects (continued)

Operation	Definition
Update	The top-level business object is modified. This operation can include adding and deleting child objects.
Delete	The top-level business object and any contained children are deleted.
Retrieve	The top-level business object and any contained children are retrieved.

For an operation that is not supported, the adapter logs the appropriate error and produces a ResourceException.

Result sets

The following table defines the operation that the adapter supports for BAPI result sets.

Table 25. Supported operation: BAPI result sets

Operation	Definition
RetrieveAll	All the matching records for the BAPI result set are retrieved.

The adapter uses the metadata information from the wrapper business object to find the operation associated with the received RFC-enabled function name. The adapter uses the application-specific information (ASI) inside the business object definition to implement the operation. After the adapter determines the operation, it sets it on the business object before sending it to the endpoint.

Supported data operations on ALE business objects

The operations that are supported by ALE business objects vary, depending on whether the business object is an outbound or inbound object. The adapter uses the application-specific information (ASI) inside the business object definition to implement the operation.

Note: Business objects generated with the ALE pass-through IDoc interface are not associated with an operation.

Outbound business objects

The operation of an ALE outbound business object is invoked by the application component that makes calls to SAP through the adapter. The adapter supports the following outbound operation.

Table 26. Supported operation: ALE outbound business objects

Operation	Definition
Execute	Posts the IDoc business object to the SAP application. This is a one-way, asynchronous operation. • If you are using the CWYAP_SAPAdapter.rar version of the adapter, no response is sent back. • If you are using the CWYAP_SAPAdapter_TX.rar version of the adapter, the transaction ID is returned.

Inbound business objects

For ALE inbound business objects, the application-specific information of an operation contains the message type, message code, and message function for an IDoc type. The adapter supports the following inbound operations.

Table 27. Supported operations: ALE inbound business objects

Operation	Definition
Create	The top-level business object and all contained children are created.
Update	The top-level business object is modified. This operation can include adding and deleting child objects.
Delete	The top-level business object and any contained children are deleted.

The adapter uses the IDoc control record field data to determine the operation that is set on the business object before sending it to the endpoint. The following fields in the control record are used to determine the operation:

- Logical_message_type (MESTYP)
- Logical_message_code (MESCOD)
- Logical_message_function(MESFCT)

Supported data operations of Query interface for SAP Software business objects

The SAP Query interface supports the RetrieveAll operation, with which you can have the results of an SAP table returned to you, and the Exists operation, which you use to determine whether data can be found in the SAP table. The adapter uses the application-specific information (ASI) inside the business object definition to implement the operation.

The supported operations for the Query interface for SAP Software are listed in the following table.

Table 28. Supported operations: Query interface for SAP Software business objects

Operation	Description
RetrieveAll	Returns a result set in the form of a container of SAP query business objects, which represent the data for each row retrieved from the table. If a table business object is sent to the SAP server (instead of a container business object), the rows are returned one at a time.
Exists	Provides a means to check for the existence of any records in SAP for a defined search criteria. The Exists operation does not return any data; it indicates whether the data exists in SAP. If no data is found, the adapter generates an exception.

Supported data operations on Advanced event processing business objects

The operations that are supported by Advanced event processing business objects vary, depending on whether the business object is an outbound or inbound object. The adapter uses the application-specific information (ASI) inside the business object definition to implement the operation.

Outbound business objects

The operation of an Advanced event processing outbound business object is invoked by the client application that makes calls to SAP through the adapter. The adapter supports the following outbound operation.

Table 29. Supported operation: Advanced event processing outbound business objects

Operation	Definition
Create	The top-level business object and all contained children are created.
Update	The top-level business object is modified. This operation can include adding and deleting child objects.
Delete	The top-level business object and any contained children are deleted.
Retrieve	The top-level business object and any contained children are retrieved.

Inbound business objects

For Advanced event processing inbound business objects, the application-specific information of an operation contains the message type, message code, and message function for an IDoc type. The adapter supports the following inbound operations.

Table 30. Supported operations: Advanced event processing inbound business objects

Operation	Definition
Create	The top-level business object and all contained children are created.
Update	The top-level business object is modified. This operation can include adding and deleting child objects.
Delete	The top-level business object and any contained children are deleted.

For WebSphere Application Server, the verb value in event table determines the operation name for AEP inbound processing.

For WebSphere Application Server, after the message is received by the endpoint, the adapter the verb value in the event table to determine the operation that is set in OutputRecord().

Naming conventions

When the J2C Bean wizard generates a business object, it provides a name for the business object that is based on the name of the corresponding business function in the SAP server. The convention applied by the SAP server when naming a business object will vary slightly depending on whether the name is for a BAPI business object, an ALE business object, an Advanced event processing business object, or a Query interface for SAP Software business object.

Naming conventions for BAPI business objects

The J2C Bean wizard provides names for the business objects for BAPIs, BAPI work unit, and BAPI result sets. At its core, the business object name reflects the structure of the business function on the SAP server.

BAPIs

When naming business objects for BAPIs, the J2C Bean wizard adds a prefix of `Sap` then converts the name of the business function to mixed case, removing any separators such as spaces or underscores, capitalizes the first letter of each word and may add an element-specific suffix (for example, `Wrapper` for top-level business object).

The following table describes the convention applied by the J2C Bean wizard when naming BAPI business objects.

Table 31. Naming conventions for BAPI business objects

Element	Naming convention
Name of the top-level business object	`Sap` + *`Name of the wrapper object you specify in the J2C Bean wizard`* + `Wrapper` For example: `SapSalesOrderWrapper`
Name of the BAPI business object	`Sap` + *`Name of the BAPI interface`* For example: `SapBapiSalesOrderCreateFromDat1` **Note:** The top-level object can contain more than one BAPI object.
Name of the child object	`Sap` + *`Name of the Structure/Table`* For example: `SapReturn`

If structures with the same name exist in different BAPIs or exist within a BAPI (for example, one at the export level and one at the table level), the J2C Bean wizard adds a unique suffix to differentiate the structures. The first structure is assigned a name (for example, `SapReturn`) and the second structure is assigned a name such as `SapReturn619647890`, where 619647890 is the unique identifier appended to the name by the J2C Bean wizard.

BAPI work unit

The following table describes the convention applied by the J2C Bean wizard when naming a BAPI work unit business object.

Table 32. Naming conventions for BAPI work unit business objects

Element	Naming convention
Name of the top-level business object	`Sap` + *`Name of the wrapper object you specify in the J2C Bean wizard`* + `Txn` For example: `SapCustomerTxn`
Name of the BAPI business object	`Sap` + *`Name of the BAPI interface`* For example: `SapCustomer`
Name of the child object	`Sap` + *`Name of the Structure/Table`* For example: `SapReturn`

If structures with the same name exist in different BAPIs or exist within a BAPI (for example, one at the export level and one at the table level), the J2C Bean wizard adds a unique suffix to differentiate the structures. The first structure is

assigned a name (for example, SapReturn) and the second structure is assigned a name such as SapReturn619647890, where 619647890 is the unique identifier appended to the name by the J2C Bean wizard.

BAPI result sets

The following table describes the convention applied by the J2C Bean wizard when naming a BAPI result sets business object.

Table 33. Naming conventions for BAPI result sets

Element	Naming convention
Name of the top-level business object	Sap + *Name of the object you specify in the J2C Bean wizard* + Resultset For example: SapCustomerGetDetailResultset
Name of the result set BAPI business object	Sap + *Name of the BAPI interface* For example: SapBapiCustomerGetDetail
Name of the child object	Sap + *Name of the Structure/Table* For example: SapReturn
Name of the query business object	Sap + *Formatted name of the query BAPI interface* For example: SapBapiCustomerGetList

If structures with the same name exist in different BAPIs or exist within a BAPI (for example, one at the export level and one at the table level), the J2C Bean wizard adds a unique suffix to differentiate the structures. The first structure is assigned a name (for example, SapReturn) and the second structure is assigned a name such as SapReturn619647890, where 619647890 is the unique identifier appended to the name by the J2C Bean wizard.

Naming conventions for ALE business objects

The J2C Bean wizard provides names for the ALE top-level business object, and the business object itself. At its core, the business object name reflects the structure of the business function on the SAP server.

Note: If you are using the ALE pass-through IDoc interface, the following naming conventions apply:

- When you select **Generic IDoc** from the Object Discovery and Selection window, the J2C Bean wizard creates a business object named SapGenericIDocObject. The naming convention described in the following sections does not apply to generic IDocs.
- When you discover an IDoc from the system or from a file, the object is named according to the naming convention for top-level wrapper objects, as described in Table 34 on page 162. No other objects are generated.

When naming business objects for ALE, the J2C Bean wizard adds a prefix of Sap then converts the name of the IDoc and extension to mixed case, removing any separators such as spaces or underscores, capitalizes the first letter of each word and may add an element-specific suffix.

The following table describes the convention applied by the J2C Bean wizard when naming ALE business objects.

Note: The *[Name of Extension type IDoc]* in the Naming convention column represents an optional entry. It is included in the name only if the selected IDoc is an Extension Type IDoc.

Table 34. Naming conventions for ALE business objects

Element	Naming convention
Name of the top-level wrapper object	Sap + *Name of IDoc* + *[Name of Extension type IDoc]* For example: SapAlereq01
Name of the IDoc business object for basic IDocs	Sap + *Name of IDoc* + BO For example, the business object for the IDoc MATMAS03 is: SapMatmas03BO
Name of the IDoc business object for extension type IDocs	Sap + *Name of IDoc* + *Name of Extension type IDoc* For example, the business object for the IDoc DELVRY03 and extension SD_DESADV_PDC is: SapDelvry03SdDesadvPdc

In the case of an IDoc duplicate name, the J2C Bean wizard adds a unique suffix to differentiate the business object. If an IDoc packet has two segments with the same name (for example segOrder), the first business object is assigned the name SapSegOrder and the second business object is assigned a name such as SapSegOrder619647890, where 619647890 is the unique identifier appended to the name by the J2C Bean wizard.

Naming conventions for Query interface for SAP Software business objects

The J2C Bean wizard provides names for the Query interface for SAP Software container, top-level business object, table object, and query object. At its core, the business object name reflects the structure of the business function on the SAP server

When naming business objects for the Query interface for SAP Software, the J2C Bean wizard adds a prefix of Sap then converts the name of the business function or SAP table to mixed case, removing any separators such as spaces or underscores, capitalizes the first letter of each word and may add an element-specific suffix (for example, Container for a container).

The following table describes the convention applied by the J2C Bean wizard when naming a Query interface for SAP Software business object.

Table 35. Naming convention for a Query interface for SAP Software business object

Element	Naming convention
Name of the container	Sap + *Name of the object you specify in the J2C Bean wizard* + Container For example: SapCustomerContainer
Name of the table object	Sap + *Name of the SAP table* For example: SapKna1
Name of the query object	Sap + *Name of the SAP table* + Querybo For example: SapKna1Querybo

Naming conventions for Advanced event processing business objects

The J2C Bean wizard provides names for Advanced event processing , top-level business object, and the business object itself. At its core, the business object name reflects the structure of the business function on the SAP server.

When naming business objects for the Advanced event processing interface, the J2C Bean wizard adds a prefix of `Sap` then converts the name of the IDoc and extension to mixed case, removing any separators such as spaces or underscores, capitalizes the first letter of each word and may add an element-specific suffix.

The following table describes the convention applied by the J2C Bean wizard when naming advanced event processing business objects.

Note: The *[Name of Extension type IDoc]* in the Naming convention column represents an optional entry. It is included in the name only if the selected IDoc is an Extension Type IDoc.

Table 36. Naming convention for Advanced event processing business objects

Element	Naming convention
Name of the top-level wrapper object	`Sap` + *Name of IDoc* + *[Name of Extension type IDoc]* For example: `SapAepreq01`
Name of the IDoc business object for basic IDocs	`Sap` + *Name of IDoc* For example, the business object for the IDoc MATMAS03 is: `SapMatmas03`
Name of the IDoc business object for extension type IDocs	`Sap` + *Name of IDoc* + *Name of Extension type IDoc* For example, the business object for the IDoc DELVRY03 and extension SD_DESADV_PDC is: `SapDelvry03SdDesadvPdc`

In the case of an IDoc duplicate name, the J2C Bean wizard adds a unique suffix to differentiate the business object. If an IDoc packet has two segments with the same name (for example segOrder), the first business object is assigned the name `SapSegOrder` and the second business object is assigned a name such as `SapSegOrder619647890`, where 619647890 is the unique identifier appended to the name by the J2C Bean wizard.

Outbound configuration properties

WebSphere Adapter for SAP Software has several categories of outbound connection configuration properties, which you set with the J2C Bean wizard while generating or creating objects and services. You can change the resource adapter and managed connection factory properties after you deploy the module to WebSphere Application Server using Rational Application Developer for WebSphere Software or the administrative console, but connection properties for the J2C Bean wizard cannot be changed after deployment.

Guide to information about properties

The properties used to configure WebSphere Adapter for SAP Software are described in detail in tables included in each of the configuration properties topics, such as Resource adapter properties, Managed connection factory properties, and so on. To help you use these tables, information about each row you might see is explained here.

The following table explains the meaning of each row that might be displayed in the table for a configuration property.

Row	Explanation
Required	A required field (property) must have a value in order for the adapter to work. Sometimes the J2C Bean wizard provides a default value for required properties. Removing a default value from a required field on the J2C Bean wizard *will not change that default value*. When a required field contains no value at all, the J2C Bean wizard will process the field using its assigned default value, and that default value will also be displayed on the administrative console. Possible values are **Yes** and **No**. Sometimes a property is required only when another property has a specific value. When this is the case, the table will note this dependency. For example, • Yes, when the EventQueryType property is set to Dynamic • Yes, for Oracle databases
Possible values	Lists and describes the possible values that you can select for the property.
Default	The predefined value that is set by the J2C Bean wizard. When the property is required, you must either accept the default value or specify one yourself. If a property has no default value, the table will state No default value. The word `None` is an acceptable default value, and does not mean that there is no default value.
Unit of measure	Specifies how the property is measured, for example in kilobytes or seconds.
Property type	Describes the property type. Valid property types include the following: • Boolean • String • Integer
Usage	Describes usage conditions or restrictions that might apply to the property. For instance, here is how a restriction would be documented: For Rational Application Developer for WebSphere Software version 6.40 or earlier, the password: • Must be uppercase • Must be 8 characters in length For versions of Rational Application Developer for WebSphere Software later than 6.40, the password: • Is not case sensitive • Can be up to 40 characters in length. This section lists other properties that affect this property or the properties that are affected by this property and describes the nature of the conditional relationship.
Example	Provides sample property values, for example: "If Language is set to JA (Japanese), Codepage number is set to `8000`".
Globalized	If a property is globalized, it has national language support, meaning that you can set the value in your national language. Valid values are **Yes** and **No**.

Row	Explanation
Bidi supported	Indicates whether the property is supported in bidirectional (bidi) processing. Bidirectional processing pertains to the task of processing data that contains both left-to-right (Hebrew or Arabic, for example) and right-to-left (a URL or file path, for example) semantic content within the same file. Valid values are **Yes** and **No**.

Connection properties for the wizard

External service connection properties establish a connection between the J2C Bean wizard of Rational Application Developer for WebSphere Software, a tool that is used to create business objects, and the SAP server. The properties you configure in the J2C Bean wizard specify such things as connection configuration, bidi properties and tracing and logging options.

Once a connection between the J2C Bean wizard and the SAP server is established, the J2C Bean wizard is able to access the metadata it needs from the SAP server to create business objects.

Some of the properties that you set in the J2C Bean wizard are used as the initial value for resource adapter, managed connection factory, and activation specification properties that you can specify at a later time in the wizard.

The external service connection properties and their purpose are described in the following table. A complete description of each property is provided in the sections that follow the table. For information on how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 163.

Note: If you set any of these connection properties using bidirectional script, you must set values that identify the format of the bidirectional script entered for that property.

Table 37. External service connection propertiesAdapter for SAP Software

Property name	Description
"Bidi direction " on page 166	The orientation component of the bidi format specification.
"Bidi ordering schema" on page 167	The ordering scheme of the bidi format specification.
"Bidi numeric shaping" on page 167	The numeric shaping component of the bid format specification.
"Bidi shaping" on page 167	The shaping component of the bidi format specification.
"Bidi symmetric swapping" on page 168	The symmetric swapping component of the bid format specification.
"Client" on page 168	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 168	Indicates the numeric identifier of the code page.
"Folder for RFC trace files" on page 169	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Host name" on page 169	Specifies the IP address or the name of the application server host that the adapter logs on to.
"Language code" on page 170	Specifies the language in which the adapter logs on.
"Log file output location property" on page 170	Specifies the location of the log file for external service.
"Logging level property" on page 170	Specifies the type error for which logging will occur during external service.

Table 37. External service connection propertiesAdapter for SAP Software (continued)

Property name	Description
"Password" on page 171	The password of the user account of the adapter on the SAP application server.
"RFC trace level" on page 172	Specifies the global trace level.
"RFC trace on" on page 172	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP interface name" on page 173	Indicates the SAP interface to be used.
"System number" on page 173	The system number of the SAP application server.
"User name" on page 174	The user account for the adapter on the SAP server.

The J2C Bean wizard uses the bidirectional connection properties to apply the proper bidirectional transformation on the data passed to the SAP server.

The bidi properties specify the bidirectional format for data coming from an external application into the adapter in the form of any business object supported by this adapter.

You should accept the default values for the bidirectional formatting properties on the J2C Bean wizard providing SAP server bidirectional format specification. When combined, these bidirectional properties define one single bidirectional format.

The default values for bidirectional formatting properties listed below are based on Windows bidirectional formatting. If the enterprise information system supports a bidirectional format other than the Windows standard bidirectional format, you must make appropriate changes to the bidi properties listed below.

Bidi direction

This property specifies the orientation component of the bidi format specification.

Table 38. Bidi direction details

Required	No
Possible values	Possible values include: • `LTR` The orientation is left-to-right • `RTL` The orientation is right-to-left • `contextualLTR` The orientation is left-to-right because of the context. A character that is not categorized as LTR that is located between two strong characters with a different writing direction, will inherit the main context's writing direction (in a LTR document the character will become LTR. • `contextualRTL` The orientation is right-to-left because of the context. A character that is not categorized as RTL that is located between two strong characters with a different writing direction, will inherit the main context's writing direction (in a RTL document the character will become RTL).
Default	`LTR`
Property type	String

Table 38. Bidi direction details (continued)

Usage	Specifies the orientation component of the bidi format specification.
Globalized	Yes
Bidi supported	No

Bidi ordering schema

This property specifies the ordering scheme of the bidi format specification.

Table 39. Bidi ordering schema details

Required	No
Possible values	`Implicit` `Visual`
Default	`Implicit`
Property type	String
Usage	Specifies the ordering scheme of the bidi format specification.
Globalized	Yes
Bidi supported	No

Bidi numeric shaping

This property specifies the numeric shaping component of the bidi format specification.

Table 40. Bidi numeric details

Required	No

Possible values	`Nominal`
	`National`
	`Contextual`

Default	`Nominal`
Property type	String
Usage	Specifies the numeric shaping component of the bidi format specification.
Globalized	Yes
Bidi supported	No

Bidi shaping

This property specifies the shaping component of the bidi format specification.

Table 41. Bidi shaping details

Required	No

Possible values	`Nominal`
	`Shaped`
	`Initial`
	`Middle`
	`Final`
	`Isolated`

Default	`Nominal`

Table 41. Bidi shaping details (continued)

Property type	String
Usage	Specifies the shaping component of the bidi format specification.
Globalized	Yes
Bidi supported	No

Bidi symmetric swapping

This property specifies the symmetric swapping component of the bidi format specification.

Table 42. Bidi symmetric swapping details

Required	No
Possible values	`True` `False`
Default	`True`
Property type	Boolean
Usage	This property specifies the symmetric swapping component of the bidi format specification.
Globalized	Yes
Bidi supported	No

Client

This property is the client number of the SAP system to which the adapter connects.

Table 43. Client details

Required	Yes
Possible values	You can enter a range of values from `000` to `999`.
Default	`100`
Property type	Integer
Usage	When an application attempts to log on to the SAP server, the SAP server requires that the application have a Client number associated with it. The Client property value identifies the client (the adapter) that is attempting to log onto the SAP server.
Globalized	No
Bidi supported	No

Codepage number

The numeric identifier of the code page.

Table 44. Codepage number details

Required	No
Possible values	You can enter a range of values from `0000` to `9999`. For a full listing of languages and associated codepage numbers supported by SAP, access SAP Note 7360.

Table 44. Codepage number details (continued)

Default	The default value for this property is conditionally determined by the value set for the **Language code** property.
Property type	Integer
Usage	The value assigned to the Codepage number defines the code page to be used and has a one-to-one relationship with the value set for the Language code property. The Codepage number establishes a connection to the appropriate language. Each language code value has a codepage number value associated with it. For example, the language code for English, is EN. If you selected EN (English) as your language code, the codepage number is automatically set to the numeric value associated with EN (English). The SAP code page number for EN (English) is `1100`.
Example	If **Language code** is set to `JA` (Japanese), **Codepage number** is set to `8000`.
Globalized	No
Bidi supported	No

Folder for RFC trace files

This property sets the fully qualified local path to the folder in which to write RFC trace files.

Table 45. Folder for RFC trace files details

Required	No
Default	No default value
Property type	String
Usage	Identifies the fully qualified local path into which RFC trace files are written. If **RFC trace on** is set to False (not selected), you are not permitted to set a value in the **Folder for RFC trace files** property.
Example	`c:\temp\rfcTraceDir`
Globalized	Yes
Bidi supported	No

Host name

Specifies the IP address or the name of the application server host that the adapter logs on to.

Table 46. Host name details

Required	Yes (when load balancing is not used).
Default	No default value
Property type	String
Usage	When the adapter is configured to run without load balancing, this property specifies the IP address or the name of the application server that the adapter logs on to.
Example	`sapServer`
Globalized	No
Bidi supported	No

Language code

SAP logon language code.

Table 47. Language code details

Required	Yes
Possible values	Each of the supported languages is preceded by a 2 character language code. The language itself is displayed in parentheses. The language codes that display in the list represent the SAP default set of 41 languages for non Unicode systems plus Arabic. For a full listing of supported language codes and languages, see the SAP documentation.
Default	The default language code will be your current locale. If your current locale is not listed as one of the supported language codes, then a default language code of `EN (English)` is used.
Property type	String
Usage	If you manually enter a language code, you do not need to enter the language in parentheses.
Example	If the system locale is English, the value for this property is `EN (English)`
Globalized	No
Bidi supported	No

Log file output location property

This property specifies the location of the log file for external service discovery.

Table 48. Log file output location details

Required	Yes
Default	The .metadata directory of the workspace.
Property type	String
Usage	Use this directory to hold the log file that will list the errors that occur during the discovery process. The type of discovery errors for which logging occurs is controlled by the **Logging level** property
Example	`C:\IBM\wid6.0\workspace\.metadata\SAPMetadataDiscovery.log`
Globalized	Yes
Bidi supported	No

Logging level property

This property specifies the type error for which logging will occur during external service.

Table 49. Logging level details

Required	No

Table 49. Logging level details (continued)

Possible values	FATAL SEVERE WARNING AUDIT INFO CONFIG DETAIL
Default	SEVERE
Property type	String
Usage	Use this property to tailor tracing capabilities. By specifying an error type, you are indicating that trace operations will occur only for errors of the type specified.
Example	Accepting the default value of SEVERE will provide trace information on errors that fall into the SEVERE category. Severe errors mean that an operation cannot continue, though the adapter can still function. Severe errors also include error conditions that indicate an impending fatal error, i.e., reporting on situations that strongly suggest that resources are on the verge of being depleted. Other error descriptions are as follows: • Fatal Adapter cannot continue. Adapter cannot function • Warning Potential error or impending error. This also includes conditions that indicate a progressive failure – for example, the potential leaking of resources. • Audit Significant event affecting adapter state or resources • Info General information outlining overall operation progress. • Config Configuration change or status. • Detail General information detailing operation progress
Globalized	Yes
Bidi supported	No

Password

This property is the password of the user account of the adapter on the SAP application server.

Table 50. Password details

Required	Yes
Default	No default value
Property type	String

Table 50. Password details (continued)

Usage	The restrictions on the password depend on the version of SAP Web Application Server. • For SAP Web Application Server version 6.40 or earlier, the password: – Must be uppercase – Must be 8 characters in length • For versions of SAP Web Application Server later than 6.40, the password: – Is not case-sensitive – Can be up to 40 characters in length
Globalized	No
Bidi supported	Yes

RFC trace level

This property specifies the global trace level.

Table 51. RFC trace level details

Required	No
Possible values	`1` - This is the default RFC trace level. When specified, SAP JCo Java API logging occurs. `3` - When specified, SAP JCo JNI API logging occurs. `5` - When specified, error diagnostic logging occurs.
Default	1
Property type	Integer
Usage	If **RFC trace on** is set to `False` (not selected), you cannot set a value in the **RFC trace level** property.
Globalized	No
Bidi supported	No

RFC trace on

This property specifies whether to generate a text file detailing the RFC activity for each event listener.

Table 52. RFC trace on details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	A value of `True` activates tracing, which generates a text file. This file is created in the directory in which the adapter process was started. The file has a prefix of rfx and a file type of trc (for example, rfc03912_02220.trc). Use these text files in a development environment only, because the files can grow rapidly. If **RFC trace on** is set to `False` (not selected), you cannot set values in the **Folder for RFC trace files** or **RFC trace level** properties.

Table 52. RFC trace on details (continued)

Example	Examples of the information in the file are RfcCall FUNCTION BAPI_CUSTOMER_GETLIST, followed by the information for the parameters in the interface, or RFC Info rfctable, followed by the data from one of the interface tables. The trace file is created in the directory where the adapter process has been started. The trace file has a .trc file extension and the file name will start with the letters rfc followed by a unique identifier. For example, `rfc03912_02220.trc`.
Globalized	No
Bidi supported	No

SAP interface name

This property indicates whether you are creating business objects for the ALE, BAPI, Advanced event processing, or Query interface for SAP Software.

Table 53. SAP interface name details

Required	Yes

Possible values	For outbound:
	`Advanced event processing (AEP)`
	`ALE`
	`ALE pass-through IDoc`
	`BAPI`
	`BAPI work unit`
	`BAPI result set`
	`Query interface for SAP Software (QSS)`
	
	For inbound:
	`Advanced event processing (AEP)`
	`ALE`
	`ALE pass-through IDoc`
	`BAPI`

Default	For outbound: `BAPI` For inbound: `ALE`
Property type	String
Usage	Specifies the interface used by the adapter. The adapter interacts with the interface to support outbound and or inbound processing by enabling the exchange of data in the form of business objects.
Globalized	No
Bidi supported	No

System number

This property is the system number of the SAP application server.

Table 54. System number details

Required	Yes
Possible values	You can enter a range of values from `00` to `99`.

Table 54. System number details (continued)

Default	00
Property type	Integer
Usage	The system number further identifies the Gateway service.
Globalized	No
Bidi supported	No

User name

This property is the user account for the adapter on the SAP server.

Table 55. User name details

Required	Yes
Default	No default value
Property type	String
Usage	Maximum length of 12 characters. The user name is not case sensitive. It is recommended that you set up a CPIC user account in the SAP application and that you give this account the necessary privileges to manipulate the data required by the business objects supported by the adapter. For example, if the adapter must perform certain SAP business transactions, the adapter's account in the SAP application must have the permissions set to allow it to perform these transactions.
Example	SapUser
Globalized	Yes
Bidi supported	Yes

Resource adapter properties

The resource adapter properties control the general operation of the adapter. You set the resource adapter properties using the J2C Bean wizard when you configure the adapter. After deploying the adapter, use the administrative console to change these properties.

The following properties for logging and tracing are deprecated:

- LogFileMaxSize
- LogFileName
- LogNumberOfFiles
- TraceFileMaxSize
- TraceFileName
- TraceNumberOfFiles

The following table lists and describes the resource adapter properties. A more detailed description of each property is provided in the sections that follow the table. For information on how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 163.

Table 56. Resource adapter properties for Adapter for SAP Software

Property name		
In the wizard	**In the administrative console**	**Description**
Adapter ID	AdapterID	Identifies the adapter instance for PMI events and for logging and tracing
Disguise user data as "XXX" in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing strings of X's instead of user data in log and trace files
(Not available)	"Enable high availability support (enableHASupport)" on page 176	Do not change this property.
(Not available)	LogFileMaxSize	Deprecated
(Not available)	LogFilename	Deprecated
(Not available)	LogNumberOfFiles	Deprecated
(Not available)	TraceFileMaxSize	Deprecated
(Not available)	TraceFileName	Deprecated
(Not available)	TraceNumberOfFiles	Deprecated

Adapter ID (AdapterID)

This property identifies a specific deployment, or instance, of the adapter.

Table 57. Adapter ID details

Required	Yes
Default	001
Property type	String
Usage	This property identifies the adapter instance in log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, SAPRA, to form the component name used by the Log and Trace Analyzer tool. For example, if the adapter ID property is set to 001, the component ID is SAPRA001. If you run multiple instances of the same adapter, make sure that the first eight characters of the adapter ID property are unique for each instance so that you can correlate log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate log and trace information to a particular instance of an adapter. To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for SAP Software to 001 and 002. The component IDs for those instances, SAPRA001 and SAPRA002, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. For example, suppose you set the adapter ID properties of two instances to Instance01 and Instance02. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to SAPRAInstance. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	Yes

Table 57. Adapter ID details (continued)

Bidi supported	No

Disguise user data as "XXX" in log and trace files (HideConfidentialTrace) property

This property specifies whether to replace user data in log and trace files with a string of X's to prevent unauthorized disclosure of potentially sensitive data.

Table 58. Disguise user data as "XXX" in log and trace files details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	If you set this property to `True`, the adapter replaces user data with a string of `X`'s when writing to log and trace files. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	No
Bidi supported	No

Enable high availability support (enableHASupport)

Do not change this property. It must be set to `true`.

Managed connection factory properties

Managed connection factory properties are used by the adapter at run time to create an outbound connection instance with the SAP server.

The following property that was specified as a Managed connection factory property in version 6.0.2.x applies to the interaction specification property group in version 6.1.x.

- IgnoreBAPIReturn

You set the managed connection factory properties using the J2C Bean wizard and can change them using the Rational Application Developer for WebSphere Software Assembly Editor, or after deployment through the WebSphere Application Server administrative console.

The following table lists and describes the managed connection factory properties. A more detailed description of each property is provided in the sections that follow the table. For information on how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 163.

Note: The J2C Bean wizard refers to these properties as managed connection factory properties and WebSphere Application Server administrative console refers to these as (J2C) connection factory properties.

Table 59. Managed connection factory properties for Adapter for SAP Software

Property name		
In the wizard	**In the administrative console**	**Description**
"ABAP debug" on page 178	ABAPDebug	ABAB debugger property.
Adapter ID	AdapterID	Identifies the adapter instance for PMI events and for logging and tracing
"Client" on page 179	Client	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 179	Codepage	Indicates the numeric identifier of the code page.
Disguise user data as "XXX" in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing strings of X's instead of user data in log and trace files
"Enable Secure Network Connection" on page 180	SncMode	Indicates whether secure network connection mode is used.
"Folder for RFC trace files" on page 181	RfcTracePath	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Gateway host" on page 181	GatewayHost	The host name of the SAP gateway.
"Gateway service" on page 182	GatewayService	The identifier of the gateway on the gateway host that carries out the RFC services.
"Host name" on page 182	ApplicationServerHost	Specifies the IP address or the name of the application server host that the adapter logs on to.
"Language code" on page 182	Language code	Specifies the Language code in which the adapter logs on to SAP.
"Message server host" on page 183	MessageServerHost	Specifies the name of the host on which the message server is running.
"Partner character set" on page 183	PartnerCharset	Specifies PartnerCharset encoding.
"Password" on page 183	Password	The password of the user account of the adapter on the SAP application server.
"RFC trace level" on page 184	RfcTraceLevel	Specifies the global trace level.
"RFC trace on" on page 184	RfcTraceOn	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP system ID" on page 185	SAPSystemID	Specifies the system ID of the SAP system for which logon load balancing is allowed.
"Secure Network Connection library path" on page 185	SncLib	Specifies the path to the library that provides the secure network connection service.
"Secure Network Connection name" on page 185	SncMyname	Specifies the name of the secure network connection.
"Secure Network Connection partner" on page 186	SncPartnername	Specifies the name of the secure network connection partner.
"Secure Network Connection security level" on page 186	SncQop	Specifies the level of security for the secure network connection.

Table 59. Managed connection factory properties for Adapter for SAP Software (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
"System number" on page 186	SystemNumber	The system number of the SAP application server.
"User name" on page 187	userName	The user account for the adapter on the SAP server.
"X509 certificate" on page 187	X509cert	Specifies the X509 certificate to be used as the logon ticket.

Adapter ID (AdapterID)

This property identifies a specific deployment, or instance, of the adapter.

Table 60. Adapter ID details

Required	Yes
Default	`001`
Property type	String
Usage	This property identifies the adapter instance in log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, `SAPRA`, to form the component name used by the Log and Trace Analyzer tool. For example, if the adapter ID property is set to `001`, the component ID is `SAPRA001`. If you run multiple instances of the same adapter, make sure that the first eight characters of the adapter ID property are unique for each instance so that you can correlate log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate log and trace information to a particular instance of an adapter. To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for SAP Software to `001` and `002`. The component IDs for those instances, `SAPRA001` and `SAPRA002`, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. For example, suppose you set the adapter ID properties of two instances to `Instance01` and `Instance02`. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to `SAPRAInstance`. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	Yes
Bidi supported	No

ABAP debug

This property specifies whether the adapter invokes the ABAP Debugger for the appropriate function module when the adapter begins processing a business object.

Table 61. ABAP debug details

Required	No

Table 61. ABAP debug details (continued)

Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	When the property is set to `True`, the adapter opens the SAP GUI in debug mode. You must have proper authorization to use the debugger. Create a dialog user ID because a CPI-C user ID cannot open an SAP GUI session. You need authorization to run in debug mode as well as any authorizations required by the ABAP code being debugged. For example, if a BAPI_CUSTOMER_CREATEFROMDATA1 is being debugged, you need authorization to create customers. You can add breakpoints only after the debugger opens. This property should always be set to `False` in a production environment. This property is supported on the Windows platform only.
Globalized	No
Bidi supported	No

Client

This property is the client number of the SAP system to which the adapter connects.

Table 62. Client details

Required	Yes
Possible values	You can enter a range of values from `000` to `999`.
Default	`100`
Property type	Integer
Usage	When an application attempts to log on to the SAP server, the SAP server requires that the application have a Client number associated with it. The Client property value identifies the client (the adapter) that is attempting to log onto the SAP server.
Globalized	No
Bidi supported	No

Codepage number

The numeric identifier of the code page.

Table 63. Codepage number details

Required	No
Possible values	You can enter a range of values from `0000` to `9999`. For a full listing of languages and associated codepage numbers supported by SAP, access SAP Note 7360.
Default	The default value for this property is conditionally determined by the value set for the **Language code** property.
Property type	Integer

Table 63. Codepage number details (continued)

Usage	The value assigned to the Codepage number defines the code page to be used and has a one-to-one relationship with the value set for the Language code property. The Codepage number establishes a connection to the appropriate language. Each language code value has a codepage number value associated with it. For example, the language code for English, is EN. If you selected EN (English) as your language code, the codepage number is automatically set to the numeric value associated with EN (English). The SAP code page number for EN (English) is `1100`.
Example	If **Language code** is set to `JA` (Japanese), **Codepage number** is set to `8000`.
Globalized	No
Bidi supported	No

Disguise user data as "XXX" in log and trace files (HideConfidentialTrace) property

This property specifies whether to replace user data in log and trace files with a string of `X`'s to prevent unauthorized disclosure of potentially sensitive data.

Table 64. Disguise user data as "XXX" in log and trace files details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	If you set this property to `True`, the adapter replaces user data with a string of `X`'s when writing to log and trace files. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	No
Bidi supported	No

Enable Secure Network Connection

This property indicates whether secure network connection mode is enabled.

Table 65. Enable Secure Network Connection details

Required	No
Possible values	`0` (off) `1` (on)
Default	`0`
Property type	String

Table 65. Enable Secure Network Connection details (continued)

Usage	Set the value to 1 (on) if you want to use secure network connection. If you set this value to 1, you must also set following properties: • "Secure Network Connection library path" on page 185 • "Secure Network Connection name" on page 185 • "Secure Network Connection partner" on page 186 • "Secure Network Connection security level" on page 186
Globalized	No
Bidi supported	No

Folder for RFC trace files

This property sets the fully qualified local path to the folder in which to write RFC trace files.

Table 66. Folder for RFC trace files details

Required	No
Default	No default value
Property type	String
Usage	Identifies the fully qualified local path into which RFC trace files are written. If **RFC trace on** is set to False (not selected), you are not permitted to set a value in the **Folder for RFC trace files** property.
Example	`c:\temp\rfcTraceDir`
Globalized	Yes
Bidi supported	No

Gateway host

This property is the Gateway host name. Enter either the IP address or the name of the Gateway host. Consult with your SAP administrator for information on the Gateway host name.

Table 67. Gateway host details

Required	Yes
Default	No default value
Property type	String
Usage	This property is the host name of the SAP gateway. The gateway enables communication between work processes on the SAP system and external programs. The host identified is used as the gateway for the resource adapter. Maximum length of 20 characters. If the computer name is longer than 20 characters, define a symbolic name in the THOSTS table.
Globalized	No
Bidi supported	No

Gateway service

This property is the identifier of the gateway on the gateway host that carries out the RFC services.

Table 68. Gateway service details

Required	Yes
Default	`sapgw00`
Property type	String
Usage	These services enable communication between work processes on the SAP server and external programs. The service typically has the format of `sapgw00`, where 00 is the SAP system number. Maximum of 20 characters.
Globalized	No
Bidi supported	No

Host name

Specifies the IP address or the name of the application server host that the adapter logs on to.

Table 69. Host name details

Required	Yes (when load balancing is not used).
Default	No default value
Property type	String
Usage	When the adapter is configured to run without load balancing, this property specifies the IP address or the name of the application server that the adapter logs on to.
Example	`sapServer`
Globalized	No
Bidi supported	No

Language code

This property specifies the Language code in which the adapter logs on.

Table 70. Language code details

Required	Yes
Possible values	For a full listing of languages and associated codepage numbers supported by SAP, access SAP Note 7360.
Default	The default value for the Language code property is based on the system locale.
Property type	String
Usage	Each of the supported languages is preceded by a 2 character language code. The language itself is displayed in parentheses. The language codes that display in the list represent the SAP default set of 41 languages for non Unicode systems plus Arabic. The value you select determines the value of the **Codepage number** property. If you manually enter a language code, you do not need to enter the language in parentheses.

Table 70. Language code details (continued)

Example	If the system locale is English, the value for this property is `EN (English)`.
Globalized	No
Bidi supported	No

Message server host

This property specifies the name of the host on which the message server is running.

Table 71. Message server host details

Required	Yes (if load balancing is used)
Default	No default value
Property type	String
Usage	This property specifies the name of the host that will inform all the servers (instances) belonging to this SAP system of the existence of the other servers to be used for load balancing. The message server host contains the information about load balancing for RFC clients so that an RFC client can be directed to an appropriate application server.
Example	`SAPERP05`
Globalized	No
Bidi supported	No

Partner character set

This property specifies the partner character set encoding.

Table 72. Partner character set details

Required	No
Default	UTF-8
Property type	String
Usage	When an encoding is specified, it is used; otherwise the default encoding is used.
Globalized	No
Bidi supported	No

Password

This property is the password of the user account of the adapter on the SAP application server.

Table 73. Password details

Required	Yes
Default	No default value
Property type	String

Table 73. Password details (continued)

Usage	The restrictions on the password depend on the version of SAP Web Application Server. • For SAP Web Application Server version 6.40 or earlier, the password: – Must be uppercase – Must be 8 characters in length • For versions of SAP Web Application Server later than 6.40, the password: – Is not case-sensitive – Can be up to 40 characters in length
Globalized	No
Bidi supported	Yes

RFC trace level

This property specifies the global trace level.

Table 74. RFC trace level details

Required	No
Possible values	`1` - This is the default RFC trace level. When specified, SAP JCo Java API logging occurs. `3` - When specified, SAP JCo JNI API logging occurs. `5` - When specified, error diagnostic logging occurs.
Default	1
Property type	Integer
Usage	If **RFC trace on** is set to `False` (not selected), you cannot set a value in the **RFC trace level** property.
Globalized	No
Bidi supported	No

RFC trace on

This property specifies whether to generate a text file detailing the RFC activity for each event listener.

Table 75. RFC trace on details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	A value of `True` activates tracing, which generates a text file. This file is created in the directory in which the adapter process was started. The file has a prefix of rfx and a file type of trc (for example, rfc03912_02220.trc). Use these text files in a development environment only, because the files can grow rapidly. If **RFC trace on** is set to `False` (not selected), you cannot set values in the **Folder for RFC trace files** or **RFC trace level** properties.

Table 75. RFC trace on details (continued)

Example	Examples of the information in the file are RfcCall FUNCTION BAPI_CUSTOMER_GETLIST, followed by the information for the parameters in the interface, or RFC Info rfctable, followed by the data from one of the interface tables. The trace file is created in the directory where the adapter process has been started. The trace file has a .trc file extension and the file name will start with the letters rfc followed by a unique identifier. For example, `rfc03912_02220.trc`.
Globalized	No
Bidi supported	No

SAP system ID

This property specifies the system ID of the SAP system for which logon load balancing is allowed.

Table 76. SAP system ID details

Required	Yes (when load balancing is used)
Default	No default value
Property type	String
Usage	Value must be three characters
Example	`DYL`
Globalized	No
Bidi supported	No

Secure Network Connection library path

This property specifies the path to the library that provides the secure network connection service.

Table 77. Secure Network Connection library path details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify the path to the library that provides the service.
Example	`/WINDOWS/system32/gssapi32.dll`
Globalized	No
Bidi supported	No

Secure Network Connection name

This property specifies the name of the secure network connection.

Table 78. Secure Network Connection name details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String

Table 78. Secure Network Connection name details (continued)

Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a name for the connection.
Example	`DOMAINNAME/USERNAME`
Globalized	No
Bidi supported	No

Secure Network Connection partner

This property specifies the name of the secure network connection partner.

Table 79. Secure Network Connection partner details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a name for the connection partner.
Example	`CN=sap00.saperpdev, OU=Adapter, O=IBM, C=US`
Globalized	No
Bidi supported	No

Secure Network Connection security level

This property specifies the level of security for the secure network connection.

Table 80. Secure Network Connection security level details

Required	Yes, if SncMode is set to 1; no otherwise.
Possible values	`1` (Authentication only) `2` (Integrity protection) `3` (Privacy protection) `8` (Use the value from `snc/data_protection/use` on the application server) `9` (Use the value from `snc/data_protection/max` on the application server)
Default	`3` (Privacy protection)
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a value to indicate the level of security for the connection.
Globalized	No
Bidi supported	No

System number

This property is the system number of the SAP application server.

Table 81. System number details

Required	Yes
Possible values	You can enter a range of values from `00` to `99`.

Table 81. System number details (continued)

Default	00
Property type	Integer
Usage	The system number further identifies the Gateway service.
Globalized	No
Bidi supported	No

User name

This property is the user account for the adapter on the SAP server.

Table 82. User name details

Required	Yes
Default	No default value
Property type	String
Usage	Maximum length of 12 characters. The user name is not case sensitive. It is recommended that you set up a CPIC user account in the SAP application and that you give this account the necessary privileges to manipulate the data required by the business objects supported by the adapter. For example, if the adapter must perform certain SAP business transactions, the adapter's account in the SAP application must have the permissions set to allow it to perform these transactions.
Example	SapUser
Globalized	Yes
Bidi supported	Yes

X509 certificate

This property specifies the X509 certificate to be used as the logon ticket.

Table 83. X509 certificate details

Required	No.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), you can provide a value for the X509 certificate.
Globalized	No
Bidi supported	No

Interaction specification properties

An interaction is an operation. Interaction specification properties control how the operation is run. The J2C Bean wizard sets the interaction specification properties when you configure the adapter.

Table 84 on page 188 lists and describes the interaction specification properties that you set. For information about how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 163.

Note: Typically, you do not need to change these properties. However, you can change some properties for outbound operations. For example, you might increase the value of the interaction specification property that specifies the maximum number of hits to be returned by a `RetrieveAll` operation, if your RetrieveAll operations do not return complete information. Use the assembly editor in Rational Application Developer for WebSphere Software to change these properties, which reside in the method binding of the import.

Table 84. Interaction specification properties for Adapter for SAP Software

Property name		Description
In the wizard	**In the assembly editor**	
"Custom retrieve function name"	customFunctionName	Indicates the name of a custom function to be used by the Query interface to SAP Software to retrieve data from an SAP table.
Function name	functionName	Populates the function name for the specific SAP interface.
Ignore errors in BAPI return	IgnoreBAPIReturn	Indicates if errors in BAPI returns will be ignored.
"Maximum number of hits for the discovery" on page 190	ResultSetLimit	Maximum number of result sets to return during a RetrieveAll operation.
"Select the queue name" on page 190	QRFCQueueName	The name of a customer-defined queue on the SAP server.

Custom retrieve function name

For the Query interface for SAP Software, this property specifies the name of a custom function that should be used to retrieve data from an SAP table.

Table 85. Custom retrieve function name details

Required	No
Default	No default value
Property type	String
Usage	This property applies to Query interface for SAP Software only. On non-Unicode systems, the default function used to retrieve data from SAP tables (RFC_READ_TABLE) might produce an exception. To avoid the problem, you can create another function on the SAP server and then indicate, during configuration, that the adapter should use this custom function to retrieve data. This property specifies the name of the custom function. **Note:** You must create the function on the SAP server before you specify this property on the J2C Bean wizard. Follow the steps listed in SAP note 758278 to make a copy of RFC_READ_TABLE and modify the copy per the note.
Globalized	No
Bidi supported	No

Function name

The `functionName` interaction specification property controls the interaction by associating operations with the proper interface.

Table 86. Function name details

Required	Yes

Table 86. Function name details (continued)

Possible values	`True` `False`
Default	Null
Property type	String

Usage	The BAPI outbound and inbound interfaces support the following values for the functionName interaction specification property:
	`WBIInteractionSpec.CREATE`
	`WBIInteractionSpec.UPDATE`
	`WBIInteractionSpec.RETRIEVE`
	`WBIInteractionSpec.DELETE`
	
	The BAPI result set supports the following values for the functionName interaction specification property:
	
	`WBIInteractionSpec.RETRIEVEALL`
	
	The ALE outbound interface supports the following value for the functionName interaction specification property:
	
	`WBIInteractionSpec.EXECUTE`
	
	The ALE inbound interface supports the following values for the functionName interaction specification property:
	
	`WBIInteractionSpec.CREATE`
	`WBIInteractionSpec.UPDATE`
	`WBIInteractionSpec.RETRIEVE`
	`WBIInteractionSpec.DELETE`
	
	The Query interface for SAP software (QISS) interface supports the following values for the functionName interaction specification property:
	• `WBIInteractionSpec.EXISTS`
	Throws exceptions `NotExistsException` and `QISSQueryFailedException`
	• `WBIInteractionSpec.RETRIEVEALL`
	Throws exceptions `QISSQueryFailedException`
	
	The Advanced event processing interface for inbound processing supports the following values for the functionName interaction specification property:
	
	`WBIInteractionSpec.CREATE`
	`WBIInteractionSpec.UPDATE`
	`WBIInteractionSpec.DELETE`
	
	The Advanced event processing interface for outbound processing supports the following values for the functionName interaction specification property:
	
	`WBIInteractionSpec.CREATE`
	`WBIInteractionSpec.UPDATE`
	`WBIInteractionSpec.RETRIEVE`
	`WBIInteractionSpec.DELETE`

Globalized	No
Bidi supported	No

Ignore errors in BAPI return

This property indicates whether to ignore errors specified in a BAPI return operation. The return structure can be data or a table.

Table 87. Ignore errors in BAPI return details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	This property applies to BAPI outbound synchronous RFC processing only. When set to `True`, the Adapter for SAP Software *ignores* the checking of error code in the BAPI RETURN structure after BAPI has run, and returns this structure to user as is. **Note:** RETURN structure is part of every BAPI and contains status of the BAPI execution. Accepting the default value of `False` results in the adapter processing the RETURN structure and throwing an exception if an error code is found.
Globalized	No
Bidi supported	No

Maximum number of hits for the discovery

For the Query interface for SAP Software, this property specifies the maximum number of result sets, which represents data for each row retrieved from a table through a RetrieveAll operation.

Table 88. Result set limit details

Required	Yes
Default	`100`
Property type	Integer
Usage	This property applies to Query interface for SAP Software only. If the number of hits in the table on the SAP server exceeds the value of the ResultSetLimit property, the adapter returns the error `MatchesExceededLimitException`. The adapter uses this property to help avoid out-of-memory issues.
Globalized	No
Bidi supported	No

Select the queue name

For BAPI outbound processing, when Asynchronous queued RFC is selected, this property specifies the name of a queue on the SAP server to which BAPIs will be delivered.

Table 89. Select the queue name details

Required	No
Default	The first queue defined on the SAP server. If no queue is defined on the SAP server, there is no default value.
Property type	String

Table 89. Select the queue name details (continued)

Usage	This property applies to BAPI outbound asynchronous queued RFC processing only. When you want to deliver BAPI calls to a queue on the SAP server, you must specify the name of the queue. During configuration, you select, from a drop-down list, an existing queue. If no queues exist on the SAP server, you can type the name of a queue.
Globalized	No
Bidi supported	No

Inbound configuration properties

WebSphere Adapter for SAP Software has several categories of inbound connection configuration properties, which you set with the J2C Bean wizard while generating or creating objects and services. You can change the resource adapter and activation specification properties after you deploy the module using Rational Application Developer for WebSphere Software or the administrative console, but connection properties for the J2C Bean wizard cannot be changed after deployment.

Guide to information about properties

The properties used to configure WebSphere Adapter for SAP Software are described in detail in tables included in each of the configuration properties topics, such as Resource adapter properties, Managed connection factory properties, and so on. To help you use these tables, information about each row you might see is explained here.

The following table explains the meaning of each row that might be displayed in the table for a configuration property.

Row	**Explanation**
Required	A required field (property) must have a value in order for the adapter to work. Sometimes the J2C Bean wizard provides a default value for required properties. Removing a default value from a required field on the J2C Bean wizard *will not change that default value*. When a required field contains no value at all, the J2C Bean wizard will process the field using its assigned default value, and that default value will also be displayed on the administrative console. Possible values are **Yes** and **No**. Sometimes a property is required only when another property has a specific value. When this is the case, the table will note this dependency. For example, • Yes, when the EventQueryType property is set to Dynamic • Yes, for Oracle databases
Possible values	Lists and describes the possible values that you can select for the property.
Default	The predefined value that is set by the J2C Bean wizard. When the property is required, you must either accept the default value or specify one yourself. If a property has no default value, the table will state No default value. The word `None` is an acceptable default value, and does not mean that there is no default value.
Unit of measure	Specifies how the property is measured, for example in kilobytes or seconds.

Row	Explanation
Property type	Describes the property type. Valid property types include the following: • Boolean • String • Integer
Usage	Describes usage conditions or restrictions that might apply to the property. For instance, here is how a restriction would be documented: For Rational Application Developer for WebSphere Software version 6.40 or earlier, the password: • Must be uppercase • Must be 8 characters in length For versions of Rational Application Developer for WebSphere Software later than 6.40, the password: • Is not case sensitive • Can be up to 40 characters in length. This section lists other properties that affect this property or the properties that are affected by this property and describes the nature of the conditional relationship.
Example	Provides sample property values, for example: "If Language is set to JA (Japanese), Codepage number is set to `8000`".
Globalized	If a property is globalized, it has national language support, meaning that you can set the value in your national language. Valid values are **Yes** and **No**.
Bidi supported	Indicates whether the property is supported in bidirectional (bidi) processing. Bidirectional processing pertains to the task of processing data that contains both left-to-right (Hebrew or Arabic, for example) and right-to-left (a URL or file path, for example) semantic content within the same file. Valid values are **Yes** and **No**.

Connection properties for the wizard

External service connection properties establish a connection between the J2C Bean wizard of Rational Application Developer for WebSphere Software, a tool that is used to create business objects, and the SAP server. The properties you configure in the J2C Bean wizard specify such things as connection configuration, bidi properties and tracing and logging options.

Once a connection between the J2C Bean wizard and the SAP server is established, the J2C Bean wizard is able to access the metadata it needs from the SAP server to create business objects.

Some of the properties that you set in the J2C Bean wizard are used as the initial value for resource adapter, managed connection factory, and activation specification properties that you can specify at a later time in the wizard.

The external service connection properties and their purpose are described in the following table. A complete description of each property is provided in the sections that follow the table. For information on how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 163.

Note: If you set any of these connection properties using bidirectional script, you must set values that identify the format of the bidirectional script entered for that property.

Table 90. External service connection propertiesAdapter for SAP Software

Property name	Description
"Bidi direction "	The orientation component of the bidi format specification.
"Bidi ordering schema" on page 194	The ordering scheme of the bidi format specification.
"Bidi numeric shaping" on page 194	The numeric shaping component of the bid format specification.
"Bidi shaping" on page 195	The shaping component of the bidi format specification.
"Bidi symmetric swapping" on page 195	The symmetric swapping component of the bid format specification.
"Client" on page 195	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 196	Indicates the numeric identifier of the code page.
"Folder for RFC trace files" on page 196	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Host name" on page 197	Specifies the IP address or the name of the application server host that the adapter logs on to.
"Language code" on page 197	Specifies the language in which the adapter logs on.
"Log file output location property" on page 197	Specifies the location of the log file for external service.
"Logging level property" on page 198	Specifies the type error for which logging will occur during external service.
"Password" on page 198	The password of the user account of the adapter on the SAP application server.
"RFC trace level" on page 199	Specifies the global trace level.
"RFC trace on" on page 199	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP interface name" on page 200	Indicates the SAP interface to be used.
"System number" on page 201	The system number of the SAP application server.
"User name" on page 201	The user account for the adapter on the SAP server.

The J2C Bean wizard uses the bidirectional connection properties to apply the proper bidirectional transformation on the data passed to the SAP server.

The bidi properties specify the bidirectional format for data coming from an external application into the adapter in the form of any business object supported by this adapter.

You should accept the default values for the bidirectional formatting properties on the J2C Bean wizard providing SAP server bidirectional format specification. When combined, these bidirectional properties define one single bidirectional format.

The default values for bidirectional formatting properties listed below are based on Windows bidirectional formatting. If the enterprise information system supports a bidirectional format other than the Windows standard bidirectional format, you must make appropriate changes to the bidi properties listed below.

Bidi direction

This property specifies the orientation component of the bidi format specification.

Table 91. Bidi direction details

Required	No
Possible values	Possible values include: • `LTR` The orientation is left-to-right • `RTL` The orientation is right-to-left • `contextualLTR` The orientation is left-to-right because of the context. A character that is not categorized as LTR that is located between two strong characters with a different writing direction, will inherit the main context's writing direction (in a LTR document the character will become LTR. • `contextualRTL` The orientation is right-to-left because of the context. A character that is not categorized as RTL that is located between two strong characters with a different writing direction, will inherit the main context's writing direction (in a RTL document the character will become RTL).
Default	`LTR`
Property type	String
Usage	Specifies the orientation component of the bidi format specification.
Globalized	Yes
Bidi supported	No

Bidi ordering schema

This property specifies the ordering scheme of the bidi format specification.

Table 92. Bidi ordering schema details

Required	No
Possible values	`Implicit` `Visual`
Default	`Implicit`
Property type	String
Usage	Specifies the ordering scheme of the bidi format specification.
Globalized	Yes
Bidi supported	No

Bidi numeric shaping

This property specifies the numeric shaping component of the bidi format specification.

Table 93. Bidi numeric details

Required	No

Possible values	`Nominal`
	`National`
	`Contextual`

Default	`Nominal`

Table 93. Bidi numeric details (continued)

Property type	String
Usage	Specifies the numeric shaping component of the bidi format specification.
Globalized	Yes
Bidi supported	No

Bidi shaping

This property specifies the shaping component of the bidi format specification.

Table 94. Bidi shaping details

Required	No

Possible values	`Nominal`
	`Shaped`
	`Initial`
	`Middle`
	`Final`
	`Isolated`

Default	`Nominal`
Property type	String
Usage	Specifies the shaping component of the bidi format specification.
Globalized	Yes
Bidi supported	No

Bidi symmetric swapping

This property specifies the symmetric swapping component of the bidi format specification.

Table 95. Bidi symmetric swapping details

Required	No
Possible values	`True` `False`
Default	`True`
Property type	Boolean
Usage	This property specifies the symmetric swapping component of the bidi format specification.
Globalized	Yes
Bidi supported	No

Client

This property is the client number of the SAP system to which the adapter connects.

Table 96. Client details

Required	Yes
Possible values	You can enter a range of values from `000` to `999`.
Default	`100`

Table 96. Client details (continued)

Property type	Integer
Usage	When an application attempts to log on to the SAP server, the SAP server requires that the application have a Client number associated with it. The Client property value identifies the client (the adapter) that is attempting to log onto the SAP server.
Globalized	No
Bidi supported	No

Codepage number

The numeric identifier of the code page.

Table 97. Codepage number details

Required	No
Possible values	You can enter a range of values from `0000` to `9999`. For a full listing of languages and associated codepage numbers supported by SAP, access SAP Note 7360.
Default	The default value for this property is conditionally determined by the value set for the **Language code** property.
Property type	Integer
Usage	The value assigned to the Codepage number defines the code page to be used and has a one-to-one relationship with the value set for the Language code property. The Codepage number establishes a connection to the appropriate language. Each language code value has a codepage number value associated with it. For example, the language code for English, is EN. If you selected EN (English) as your language code, the codepage number is automatically set to the numeric value associated with EN (English). The SAP code page number for EN (English) is `1100`.
Example	If **Language code** is set to `JA` (Japanese), **Codepage number** is set to `8000`.
Globalized	No
Bidi supported	No

Folder for RFC trace files

This property sets the fully qualified local path to the folder in which to write RFC trace files.

Table 98. Folder for RFC trace files details

Required	No
Default	No default value
Property type	String
Usage	Identifies the fully qualified local path into which RFC trace files are written. If **RFC trace on** is set to False (not selected), you are not permitted to set a value in the **Folder for RFC trace files** property.
Example	`c:\temp\rfcTraceDir`
Globalized	Yes
Bidi supported	No

Host name

Specifies the IP address or the name of the application server host that the adapter logs on to.

Table 99. Host name details

Required	Yes (when load balancing is not used).
Default	No default value
Property type	String
Usage	When the adapter is configured to run without load balancing, this property specifies the IP address or the name of the application server that the adapter logs on to.
Example	`sapServer`
Globalized	No
Bidi supported	No

Language code

SAP logon language code.

Table 100. Language code details

Required	Yes
Possible values	Each of the supported languages is preceded by a 2 character language code. The language itself is displayed in parentheses. The language codes that display in the list represent the SAP default set of 41 languages for non Unicode systems plus Arabic. For a full listing of supported language codes and languages, see the SAP documentation.
Default	The default language code will be your current locale. If your current locale is not listed as one of the supported language codes, then a default language code of `EN (English)` is used.
Property type	String
Usage	If you manually enter a language code, you do not need to enter the language in parentheses.
Example	If the system locale is English, the value for this property is `EN (English)`
Globalized	No
Bidi supported	No

Log file output location property

This property specifies the location of the log file for external service discovery.

Table 101. Log file output location details

Required	Yes
Default	The .metadata directory of the workspace.
Property type	String
Usage	Use this directory to hold the log file that will list the errors that occur during the discovery process. The type of discovery errors for which logging occurs is controlled by the **Logging level** property

Table 101. Log file output location details (continued)

Example	`C:\IBM\wid6.0\workspace\.metadata\SAPMetadataDiscovery.log`
Globalized	Yes
Bidi supported	No

Logging level property

This property specifies the type error for which logging will occur during external service.

Table 102. Logging level details

Required	No

Possible values	`FATAL`
	`SEVERE`
	`WARNING`
	`AUDIT`
	`INFO`
	`CONFIG`
	`DETAIL`

Default	`SEVERE`
Property type	String
Usage	Use this property to tailor tracing capabilities. By specifying an error type, you are indicating that trace operations will occur only for errors of the type specified.
Example	Accepting the default value of SEVERE will provide trace information on errors that fall into the SEVERE category. Severe errors mean that an operation cannot continue, though the adapter can still function. Severe errors also include error conditions that indicate an impending fatal error, i.e., reporting on situations that strongly suggest that resources are on the verge of being depleted. Other error descriptions are as follows: • Fatal Adapter cannot continue. Adapter cannot function • Warning Potential error or impending error. This also includes conditions that indicate a progressive failure – for example, the potential leaking of resources. • Audit Significant event affecting adapter state or resources • Info General information outlining overall operation progress. • Config Configuration change or status. • Detail General information detailing operation progress
Globalized	Yes
Bidi supported	No

Password

This property is the password of the user account of the adapter on the SAP application server.

Table 103. Password details

Required	Yes
Default	No default value
Property type	String
Usage	The restrictions on the password depend on the version of SAP Web Application Server. • For SAP Web Application Server version 6.40 or earlier, the password: – Must be uppercase – Must be 8 characters in length • For versions of SAP Web Application Server later than 6.40, the password: – Is not case-sensitive – Can be up to 40 characters in length
Globalized	No
Bidi supported	Yes

RFC trace level

This property specifies the global trace level.

Table 104. RFC trace level details

Required	No
Possible values	`1` - This is the default RFC trace level. When specified, SAP JCo Java API logging occurs. `3` - When specified, SAP JCo JNI API logging occurs. `5` - When specified, error diagnostic logging occurs.
Default	`1`
Property type	Integer
Usage	If **RFC trace on** is set to `False` (not selected), you cannot set a value in the **RFC trace level** property.
Globalized	No
Bidi supported	No

RFC trace on

This property specifies whether to generate a text file detailing the RFC activity for each event listener.

Table 105. RFC trace on details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean

Table 105. RFC trace on details (continued)

Usage	A value of `True` activates tracing, which generates a text file. This file is created in the directory in which the adapter process was started. The file has a prefix of rfx and a file type of trc (for example, rfc03912_02220.trc). Use these text files in a development environment only, because the files can grow rapidly. If **RFC trace on** is set to `False` (not selected), you cannot set values in the **Folder for RFC trace files** or **RFC trace level** properties.
Example	Examples of the information in the file are RfcCall FUNCTION BAPI_CUSTOMER_GETLIST, followed by the information for the parameters in the interface, or RFC Info rfctable, followed by the data from one of the interface tables. The trace file is created in the directory where the adapter process has been started. The trace file has a .trc file extension and the file name will start with the letters rfc followed by a unique identifier. For example, `rfc03912_02220.trc`.
Globalized	No
Bidi supported	No

SAP interface name

This property indicates whether you are creating business objects for the ALE, BAPI, Advanced event processing, or Query interface for SAP Software.

Table 106. SAP interface name details

Required	Yes

Possible values	For outbound:
	`Advanced event processing (AEP)`
	`ALE`
	`ALE pass-through IDoc`
	`BAPI`
	`BAPI work unit`
	`BAPI result set`
	`Query interface for SAP Software (QSS)`
	
	For inbound:
	`Advanced event processing (AEP)`
	`ALE`
	`ALE pass-through IDoc`
	`BAPI`

Default	For outbound: `BAPI` For inbound: `ALE`
Property type	String
Usage	Specifies the interface used by the adapter. The adapter interacts with the interface to support outbound and or inbound processing by enabling the exchange of data in the form of business objects.
Globalized	No
Bidi supported	No

System number

This property is the system number of the SAP application server.

Table 107. System number details

Required	Yes
Possible values	You can enter a range of values from `00` to `99`.
Default	`00`
Property type	Integer
Usage	The system number further identifies the Gateway service.
Globalized	No
Bidi supported	No

User name

This property is the user account for the adapter on the SAP server.

Table 108. User name details

Required	Yes
Default	No default value
Property type	String
Usage	Maximum length of 12 characters. The user name is not case sensitive. It is recommended that you set up a CPIC user account in the SAP application and that you give this account the necessary privileges to manipulate the data required by the business objects supported by the adapter. For example, if the adapter must perform certain SAP business transactions, the adapter's account in the SAP application must have the permissions set to allow it to perform these transactions.
Example	`SapUser`
Globalized	Yes
Bidi supported	Yes

Resource adapter properties

The resource adapter properties control the general operation of the adapter. You set the resource adapter properties using the J2C Bean wizard when you configure the adapter. After deploying the adapter, use the administrative console to change these properties.

The following properties for logging and tracing are deprecated:

- LogFileMaxSize
- LogFileName
- LogNumberOfFiles
- TraceFileMaxSize
- TraceFileName
- TraceNumberOfFiles

The following table lists and describes the resource adapter properties. A more detailed description of each property is provided in the sections that follow the

table. For information on how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 163.

Table 109. Resource adapter properties for Adapter for SAP Software

Property name		
In the wizard	**In the administrative console**	**Description**
Adapter ID	AdapterID	Identifies the adapter instance for PMI events and for logging and tracing
Disguise user data as ″XXX″ in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing strings of X's instead of user data in log and trace files
(Not available)	"Enable high availability support (enableHASupport)" on page 203	Do not change this property.
(Not available)	LogFileMaxSize	Deprecated
(Not available)	LogFilename	Deprecated
(Not available)	LogNumberOfFiles	Deprecated
(Not available)	TraceFileMaxSize	Deprecated
(Not available)	TraceFileName	Deprecated
(Not available)	TraceNumberOfFiles	Deprecated

Adapter ID (AdapterID)

This property identifies a specific deployment, or instance, of the adapter.

Table 110. Adapter ID details

Required	Yes
Default	001
Property type	String

Table 110. Adapter ID details (continued)

Usage	This property identifies the adapter instance in log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, `SAPRA`, to form the component name used by the Log and Trace Analyzer tool. For example, if the adapter ID property is set to `001`, the component ID is `SAPRA001`. If you run multiple instances of the same adapter, make sure that the first eight characters of the adapter ID property are unique for each instance so that you can correlate log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate log and trace information to a particular instance of an adapter. To illustrate how the length of the adapter ID property affects the filtering of log and trace files, suppose you set the adapter ID property of two instances of WebSphere Adapter for SAP Software to `001` and `002`. The component IDs for those instances, `SAPRA001` and `SAPRA002`, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. For example, suppose you set the adapter ID properties of two instances to `Instance01` and `Instance02`. You will not be able to separately examine the log and trace information for each adapter instance because the component ID for both instances is truncated to `SAPRAInstance`. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	Yes
Bidi supported	No

Disguise user data as "XXX" in log and trace files (HideConfidentialTrace) property

This property specifies whether to replace user data in log and trace files with a string of `X`'s to prevent unauthorized disclosure of potentially sensitive data.

Table 111. Disguise user data as "XXX" in log and trace files details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	If you set this property to `True`, the adapter replaces user data with a string of `X`'s when writing to log and trace files. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, can be set both at the resource adapter level and the managed connection factory level. After using the J2C Bean wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently.
Globalized	No
Bidi supported	No

Enable high availability support (enableHASupport)

Do not change this property. It must be set to `true`.

Activation specification properties for BAPI inbound processing

Activation specification properties hold the inbound event processing configuration information for a message endpoint.

Activation specification properties are used during endpoint activation to notify the adapter of eligible event listeners. During inbound processing, the adapter uses these event listeners to receive events before forwarding them to the endpoint.

You set the activation specification properties using the J2C Bean wizard and can change them using the Rational Application Developer for WebSphere Software Assembly Editor, or after deployment through the WebSphere Application Server administrative console.

Table 112 lists and describes the activation specification properties that apply to both synchronous RFC and asynchronous transactional RFC. Table 113 on page 205 applies only to asynchronous transaction RFC properties that are used for assured-once delivery.

A more detailed description of each property is provided in the sections that follow the tables. For information on how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 163.

Table 112. Activation specification properties for BAPI inbound processing

Property name		
In the wizard	**In the administrative console**	**Description**
"Client" on page 207	Client	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 207	Codepage	Indicates the numeric identifier of the code page.
"Enable Security Network Connection" on page 208	SncMode	Indicates whether secure network connection mode is used.
Retry limit for failed events	FailedEventRetryLimit	The number of times the adapter attempts to redeliver an event before marking the event as failed
"Folder for RFC trace files" on page 210	RfcTracePath	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Gateway host" on page 210	GatewayHost	The host name of the SAP gateway.
"Gateway service" on page 211	GatewayService	The identifier of the gateway on the gateway host that carries out the RFC services.
"Host name" on page 211	ApplicationServerHost	Specifies the IP address or the name of the application server host that the adapter logs on to.
"Language code" on page 211	Language code	Specifies the Language code in which the adapter logs on to SAP.
"Logon group name" on page 212	Group	An identifier of the name of the group of application server instances that have been defined in transaction SMLG and linked together for logon load balancing.
"Maximum number of retries in case of system connection failure" on page 212	retryLimit	Specifies the number of times the adapter tries to restart the event listeners.

Table 112. Activation specification properties for BAPI inbound processing (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
"Message server host" on page 213	MessageServerHost	Specifies the name of the host on which the message server is running.
"Number of listeners" on page 213	NumberOfListeners	Specifies the number of event listeners that are to be started.
"Partner character set" on page 213	PartnerCharset	Specifies PartnerCharset encoding.
"Password" on page 214	Password	The password of the user account of the adapter on the SAP application server.
"RFC program ID" on page 215	RfcProgramID	The remote function call identifier under which the adapter registers in the SAP gateway.
"RFC trace level" on page 215	RfcTraceLevel	Specifies the global trace level.
"RFC trace on" on page 215	RfcTraceOn	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP system ID" on page 216	SAPSystemID	Specifies the system ID of the SAP system for which logon load balancing is allowed.
"Secure Network Connection library path" on page 216	SncLib	Specifies the path to the library that provides the secure network connection service.
"Secure Network Connection name" on page 217	SncMyname	Specifies the name of the secure network connection.
"Secure Network Connection partner" on page 217	SncPartnername	Specifies the name of the secure network connection partner.
"Secure Network Connection security level" on page 217	SncQop	Specifies the level of security for the secure network connection.
"System number" on page 218	SystemNumber	The system number of the SAP application server.
"Time between retries in case of system connection failure (milliseconds)" on page 218	retryInterval	Specifies the time interval between attempts to restart the event listeners.
"User name" on page 218	userName	The user account for the adapter on the SAP server.
"X509 certificate" on page 219	X509cert	Specifies the X509 certificate to be used as the logon ticket.

The properties in the following table applies only to assured-once delivery. When you select assured-once delivery, the transaction ID sent from the SAP server is stored in a data source. You specify information about the data source with these properties.

Table 113. Additional activation specification properties for assured-once delivery

Property name		
In the wizard	**In the administrative console**	**Description**
"Assured once-only delivery " on page 206	AssuredOnceDelivery	Specifies whether to provide assured-once delivery for inbound events.

Table 113. Additional activation specification properties for assured-once delivery (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
"Auto create event table"	EP_CreateTable	Indicates whether the adapter should create the event recovery table automatically if it does not already exist.
"Event recovery data source (JNDI) name" on page 209	EP_SchemaName	The schema used for automatically creating the event recovery table.
"Event recovery data source (JNDI) name" on page 209	EP_DataSource_JNDIName	The JNDI name of the data source configured for event recovery.
"Event recovery table name" on page 209	EP_TableName	The name of the event recovery table.
"Password used to connect to event data source" on page 214	EP_Password	The user password for connecting to the database.
"User name used to connect to event data source" on page 219	EP_UserName	The user name for connecting to the database.

Assured once-only delivery

This property specifies whether to provide assured once-only delivery for inbound events.

Table 114. Assured once-only delivery details

Required	No
Default	`False`
Property type	Boolean
Usage	When this property is set to `True`, the adapter provides assured once event delivery. This means that each event will be delivered **once and only once**. A value of `False` does not provide assured once event delivery, but provides better performance. When this property is set to `True`, the adapter attempts to store transaction (XID) information in the event store. If it is set to `False`, the adapter does not attempt to store the information. This property is used only if the export component is transactional. If the export component is not transactional, no transaction can be used, regardless of the value of this property.
Globalized	No
Bidi supported	No

Note: The **Assured once-only delivery** property applies only to asynchronous transactional RFC processing.

Auto create event table

Determines if the event table is created automatically.

Table 115. Auto create event table details

Required	Yes, if **Assured once-only event delivery** is set to `True`, No otherwise.
Possible values	`True` `False`

Table 115. Auto create event table details (continued)

Default	`True`
Property type	Boolean
Usage	This property indicates whether the adapter should create the event recovery table automatically if it does not already exist. In the administrative console, this property is listed as ″EP_CreateTable″. If you specify a value of `True` to automatically create the table, you must specify information about the event table (such as the event recovery table name). The value provided in the Event recovery table name property is used to create the table.
Globalized	No
Bidi supported	No

Note: The **Auto create event table** property applies only to asynchronous transactional RFC processing.

Client

This property is the client number of the SAP system to which the adapter connects.

Table 116. Client details

Required	Yes
Possible values	You can enter a range of values from `000` to `999`.
Default	`100`
Property type	Integer
Usage	When an application attempts to log on to the SAP server, the SAP server requires that the application have a Client number associated with it. The Client property value identifies the client (the adapter) that is attempting to log onto the SAP server.
Globalized	No
Bidi supported	No

Codepage number

The numeric identifier of the code page.

Table 117. Codepage number details

Required	No
Possible values	You can enter a range of values from `0000` to `9999`. For a full listing of languages and associated codepage numbers supported by SAP, access SAP Note 7360.
Default	The default value for this property is conditionally determined by the value set for the **Language code** property.
Property type	Integer

Table 117. Codepage number details (continued)

Usage	The value assigned to the Codepage number defines the code page to be used and has a one-to-one relationship with the value set for the Language code property. The Codepage number establishes a connection to the appropriate language. Each language code value has a codepage number value associated with it. For example, the language code for English, is EN. If you selected EN (English) as your language code, the codepage number is automatically set to the numeric value associated with EN (English). The SAP code page number for EN (English) is `1100`.
Example	If **Language code** is set to `JA` (Japanese), **Codepage number** is set to `8000`.
Globalized	No
Bidi supported	No

Database schema name

This property is the schema used for automatically creating the event recovery table.

Note: In the administrative console, this property is listed as ″EP_SchemaName″.

Table 118. Database schema name details

Required	No
Default	No default value.
Property type	String
Usage	Specifies the schema name for the database used by the adapters event persistence feature.
Example	`ALE_SCHEMA`
Globalized	Yes
Bidi supported	No

Note: The **Database schema name** property applies only to asynchronous transactional RFC processing.

Enable Security Network Connection

This property indicates whether secure network connection mode is enabled.

Table 119. Enable Security Network Connection details

Required	No
Possible values	`0` (off) `1` (on)
Default	`0`
Property type	String
Usage	Set the value to 1 (on) if you want to use secure network connection. If you set this value to 1, you must also set following properties: • “Secure Network Connection library path” on page 216 • “Secure Network Connection name” on page 217 • “Secure Network Connection partner” on page 217 • “Secure Network Connection security level” on page 217

Table 119. Enable Security Network Connection details (continued)

Globalized	No
Bidi supported	No

Event recovery data source (JNDI) name

This property is the JNDI name of the data source configured for event recovery.

Note: In the administrative console, this property is listed as "EP_DataSource_JNDIName".

Table 120. Event recovery data source (JNDI) name details

Required	Yes
Default	No default value.
Property type	String
Usage	Used in event recovery processing. The data source must be created in WebSphere Process Server. The adapter utilizes data source for *persisting* the event state.
Example	`jdbc/DB2`
Globalized	No
Bidi supported	No

Note: The **Event recovery data source (JNDI) name** property applies only to asynchronous transactional RFC processing.

Event recovery table name

This property is the name of the event recovery table.

Note: In the administrative console, this property is listed as "EP_TableName".

Table 121. Event recovery table name details

Required	Yes
Default	No default value.
Property type	String
Usage	Used in event recovery processing. Consult database documentation for information on naming conventions. It is recommended that a separate event recovery table is configured for each endpoint. The same data source can be used to hold all of the event recovery tables.
Example	`EVENT_TABLE`
Globalized	No
Bidi supported	No

Note: The **Event recovery table name** property applies only to asynchronous transactional RFC processing.

Retry limit for failed events (FailedEventRetryLimit)

This property specifies the number of times that the adapter attempts to redeliver an event before marking the event as failed.

Table 122. Retry limit for failed events details

Required	No
Possible values	Integers
Default	5
Property type	Integer
Usage	Use this property to control how many times the adapter tries to send an event before marking it as failed. It accepts the following values: **Default** If this property is not set, the adapter tries five additional times before marking the event as failed. **0** The adapter tries to deliver the event an infinite number of times. When the property is set to 0, the event remains in the event store and the event is never marked as failed. **> 0** For integers greater than zero, the adapter retries the specified number of times before marking the event as failed. **< 0** For negative integers, the adapter does not retry failed events.
Globalized	No
Bidi supported	No

Folder for RFC trace files

This property sets the fully qualified local path to the folder in which to write RFC trace files.

Table 123. Folder for RFC trace files details

Required	No
Default	No default value
Property type	String
Usage	Identifies the fully qualified local path into which RFC trace files are written. If **RFC trace on** is set to False (not selected), you are not permitted to set a value in the **Folder for RFC trace files** property.
Example	`c:\temp\rfcTraceDir`
Globalized	Yes
Bidi supported	No

Gateway host

This property is the Gateway host name. Enter either the IP address or the name of the Gateway host. Consult with your SAP administrator for information on the Gateway host name.

Table 124. Gateway host details

Required	Yes

Table 124. Gateway host details (continued)

Default	No default value
Property type	String
Usage	This property is the host name of the SAP gateway. The gateway enables communication between work processes on the SAP system and external programs. The host identified is used as the gateway for the resource adapter. Maximum length of 20 characters. If the computer name is longer than 20 characters, define a symbolic name in the THOSTS table.
Globalized	No
Bidi supported	No

Gateway service

This property is the identifier of the gateway on the gateway host that carries out the RFC services.

Table 125. Gateway service details

Required	Yes
Default	`sapgw00`
Property type	String
Usage	These services enable communication between work processes on the SAP server and external programs. The service typically has the format of `sapgw00`, where 00 is the SAP system number. Maximum of 20 characters.
Globalized	No
Bidi supported	No

Host name

Specifies the IP address or the name of the application server host that the adapter logs on to.

Table 126. Host name details

Required	Yes (when load balancing is not used).
Default	No default value
Property type	String
Usage	When the adapter is configured to run without load balancing, this property specifies the IP address or the name of the application server that the adapter logs on to.
Example	`sapServer`
Globalized	No
Bidi supported	No

Language code

This property specifies the Language code in which the adapter logs on.

Table 127. Language code details

Required	Yes
Possible values	For a full listing of languages and associated codepage numbers supported by SAP, access SAP Note 7360.
Default	The default value for the Language code property is based on the system locale.
Property type	String
Usage	Each of the supported languages is preceded by a 2 character language code. The language itself is displayed in parentheses. The language codes that display in the list represent the SAP default set of 41 languages for non Unicode systems plus Arabic. The value you select determines the value of the **Codepage number** property. If you manually enter a language code, you do not need to enter the language in parentheses.
Example	If the system locale is English, the value for this property is `EN (English)`.
Globalized	No
Bidi supported	No

Logon group name

This property is an identifier for the name of the group of application server instances that have been defined in transaction SMLG and linked together for logon load balancing.

Table 128. Logon group details

Required	Yes (if load balancing is used)
Possible values	Consult SAP documentation for information on creating Logon groups and on calling transaction SMLG.
Default	No default value
Property type	String
Usage	When the adapter is configured for load balancing, this property represents the name of the group of application server instances that have been defined in transaction SMLG and linked together for logon load balancing. Logon load balancing allows for the dynamic distribution of logon connections to application server instances. Maximum of 20 characters. On most SAP systems, the SPACE logon group is reserved by SAP.
Globalized	No
Bidi supported	No

Maximum number of retries in case of system connection failure

This property specifies the number of times the adapter tries to restart the event listeners.

Table 129. Maximum number of retries in case of system failure details

Required	Yes
Default	`0`
Property type	Integer

Table 129. Maximum number of retries in case of system failure details (continued)

Usage	When the adapter encounters an error related to the inbound connection (if the SAP application is down for example), this property specifies the number of times the adapter tries to restart the event listeners. A value of 0 indicates an infinite number of retries. **Note:** Configure the **Time between retries in case of system connection failure (milliseconds)** appropriately when retrying infinitely. For each retry attempt, the adapter waits based on the time interval specified in the **Time between retries in case of system connection failure (milliseconds)**. **Note:** If all the retry attempts fail, the adapter logs relevant messages and CEI events and stops attempting to recover the event listener. If you reach this point, you may need to restart the application manually.
Globalized	No
Bidi supported	No

Message server host

This property specifies the name of the host on which the message server is running.

Table 130. Message server host details

Required	Yes (if load balancing is used)
Default	No default value
Property type	String
Usage	This property specifies the name of the host that will inform all the servers (instances) belonging to this SAP system of the existence of the other servers to be used for load balancing. The message server host contains the information about load balancing for RFC clients so that an RFC client can be directed to an appropriate application server.
Example	`SAPERP05`
Globalized	No
Bidi supported	No

Number of listeners

This property specifies the number of listeners that are started by an event.

Table 131. Number of listeners details

Required	No
Default	1
Property type	Integer
Usage	For event sequencing, this property should be set to 1. To improve adapter performance, you can increase the number of listeners.
Globalized	No
Bidi supported	No

Partner character set

This property specifies the partner character set encoding.

Table 132. Partner character set details

Required	No
Default	UTF-8
Property type	String
Usage	When an encoding is specified, it is used; otherwise the default encoding is used.
Globalized	No
Bidi supported	No

Password

This property is the password of the user account of the adapter on the SAP application server.

Table 133. Password details

Required	Yes
Default	No default value
Property type	String
Usage	The restrictions on the password depend on the version of SAP Web Application Server. • For SAP Web Application Server version 6.40 or earlier, the password: – Must be uppercase – Must be 8 characters in length • For versions of SAP Web Application Server later than 6.40, the password: – Is not case-sensitive – Can be up to 40 characters in length
Globalized	No
Bidi supported	Yes

Password used to connect to event data source

This property is the user password for connecting to the database.

Note: In the administrative console, this property is listed as ″EP_Password″.

Table 134. Password to connect to event data source details

Required	Yes
Default	No default value.
Property type	String
Usage	This property specifies the password used by event persistence processing to obtain the database connection from the data source.
Globalized	Yes
Bidi supported	No

Note: The **Password used to connect to event data source** property applies only to asynchronous transactional RFC processing.

RFC program ID

This property is the program identifier under which the adapter registers in the SAP gateway.

Table 135. RFC program ID details

Required	Yes
Possible values	Use the SAP transaction SM59 (Display and Maintain RFC Destinations) to see a list of available RFC program IDs.
Default	No default value.
Property type	String
Usage	The adapter registers with the gateway so that listener threads can process events from RFC-enabled functions. This value must match the program ID registered in the SAP application. The maximum length is 64 characters.
Globalized	No
Bidi supported	No

RFC trace level

This property specifies the global trace level.

Table 136. RFC trace level details

Required	No
Possible values	`1` - This is the default RFC trace level. When specified, SAP JCo Java API logging occurs. `3` - When specified, SAP JCo JNI API logging occurs. `5` - When specified, error diagnostic logging occurs.
Default	`1`
Property type	Integer
Usage	If **RFC trace on** is set to `False` (not selected), you cannot set a value in the **RFC trace level** property.
Globalized	No
Bidi supported	No

RFC trace on

This property specifies whether to generate a text file detailing the RFC activity for each event listener.

Table 137. RFC trace on details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean

Table 137. RFC trace on details (continued)

Usage	A value of `True` activates tracing, which generates a text file. This file is created in the directory in which the adapter process was started. The file has a prefix of rfx and a file type of trc (for example, rfc03912_02220.trc). Use these text files in a development environment only, because the files can grow rapidly. If **RFC trace on** is set to `False` (not selected), you cannot set values in the **Folder for RFC trace files** or **RFC trace level** properties.
Example	Examples of the information in the file are RfcCall FUNCTION BAPI_CUSTOMER_GETLIST, followed by the information for the parameters in the interface, or RFC Info rfctable, followed by the data from one of the interface tables. The trace file is created in the directory where the adapter process has been started. The trace file has a .trc file extension and the file name will start with the letters rfc followed by a unique identifier. For example, `rfc03912_02220.trc`.
Globalized	No
Bidi supported	No

SAP system ID

This property specifies the system ID of the SAP system for which logon load balancing is allowed.

Table 138. SAP system ID details

Required	Yes (when load balancing is used)
Default	No default value
Property type	String
Usage	Value must be three characters
Example	`DYL`
Globalized	No
Bidi supported	No

Secure Network Connection library path

This property specifies the path to the library that provides the secure network connection service.

Table 139. Secure Network Connection library path details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify the path to the library that provides the service.
Example	`/WINDOWS/system32/gssapi32.dll`
Globalized	No
Bidi supported	No

Secure Network Connection name

This property specifies the name of the secure network connection.

Table 140. Secure Network Connection name details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a name for the connection.
Example	`DOMAINNAME/USERNAME`
Globalized	No
Bidi supported	No

Secure Network Connection partner

This property specifies the name of the secure network connection partner.

Table 141. Secure Network Connection partner details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a name for the connection partner.
Example	`CN=sap00.saperpdev, OU=Adapter, O=IBM, C=US`
Globalized	No
Bidi supported	No

Secure Network Connection security level

This property specifies the level of security for the secure network connection.

Table 142. Secure Network Connection security level details

Required	Yes, if SncMode is set to 1; no otherwise.
Possible values	`1` (Authentication only) `2` (Integrity protection) `3` (Privacy protection) `8` (Use the value from `snc/data_protection/use` on the application server) `9` (Use the value from `snc/data_protection/max` on the application server)
Default	`3` (Privacy protection)
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a value to indicate the level of security for the connection.
Globalized	No
Bidi supported	No

System number

This property is the system number of the SAP application server.

Table 143. System number details

Required	Yes
Possible values	You can enter a range of values from `00` to `99`.
Default	`00`
Property type	Integer
Usage	The system number further identifies the Gateway service.
Globalized	No
Bidi supported	No

Time between retries in case of system connection failure (milliseconds)

This property specifies the time interval between attempts to restart the event listeners.

Table 144. Time between retries in case of system connection failure details

Required	Yes
Default	`60000`
Unit of measure	Milliseconds
Property type	Integer
Usage	When the adapter encounters an error related to the inbound connection, this property specifies the time interval the adapter waits in between attempts to restart the event listeners.
Globalized	No
Bidi supported	No

User name

This property is the user account for the adapter on the SAP server.

Table 145. User name details

Required	Yes
Default	No default value
Property type	String
Usage	Maximum length of 12 characters. The user name is not case sensitive. It is recommended that you set up a CPIC user account in the SAP application and that you give this account the necessary privileges to manipulate the data required by the business objects supported by the adapter. For example, if the adapter must perform certain SAP business transactions, the adapter's account in the SAP application must have the permissions set to allow it to perform these transactions.
Example	`SapUser`
Globalized	Yes
Bidi supported	Yes

User name used to connect to event data source

This property is the user name for connecting to the database.

Note: In the administrative console, this property is listed as ″EP_UserName″.

Table 146. User name to connect to event data source details

Required	Yes
Default	No default value.
Property type	String
Usage	User name used by event persistence for getting the database connection from the data source. Consult database documentation for information on naming conventions.
Globalized	Yes
Bidi supported	No

Note: The **User name used to connect to event data source** property applies only to asynchronous transactional RFC processing.

X509 certificate

This property specifies the X509 certificate to be used as the logon ticket.

Table 147. X509 certificate details

Required	No.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), you can provide a value for the X509 certificate.
Globalized	No
Bidi supported	No

Activation specification properties for ALE inbound processing

Activation specification properties hold the inbound event processing configuration information for a message endpoint.

Activation specification properties are used during endpoint activation to notify the adapter of eligible event listeners. During inbound processing, the adapter uses these event listeners to receive events before forwarding them to the endpoint.

You set the activation specification properties using the J2C Bean wizard and can change them using the Rational Application Developer for WebSphere Software Assembly Editor, or after deployment through the WebSphere Application Server administrative console.

The following table lists and describes the activation specification properties for ALE inbound processing. A more detailed description of each property is provided in the sections that follow the table. For information on how to read the property detail tables in the sections that follow, see “Guide to information about properties” on page 163.

Table 148. Activation specification properties for ALE inbound processing

Property name		
In the wizard	**In the administrative console**	**Description**
"ALE failure code" on page 222	AleFailureCode	Specifies the status code for dispatch failure.
"ALE failure text" on page 222	AleFailureText	Specifies the descriptive text for dispatch failure.
"ALE selective update" on page 222	AleSelectiveUpdate	Specifies which IDoc Type and MessageType combinations are to be updated when the adapter is configured to update a standard SAP status code.
"ALE status message code" on page 223	AleStatusMsgCode	If required, specifies the message code to use when the adapter posts the ALEAUD Message IDoc (ALEAUD01).
"ALE success code" on page 223	AleSuccessCode	Specifies the success status code for Application Document Posted.
"ALE success text" on page 224	AleSuccessText	Specifies the descriptive text for successful Application Document Posted.
"ALE update status" on page 224	AleUpdateStatus	Specifies whether an audit trail is required for all message types.
"Assured once-only delivery " on page 225	AssuredOnceDelivery	Specifies whether to provide assured-once delivery for inbound events.
"Auto create event table" on page 225	EP_CreateTable	Indicates whether the adapter should create the event recovery table automatically if it does not already exist.
"Client" on page 226	Client	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 226	Codepage	Indicates the numeric identifier of the code page.
"Event recovery data source (JNDI) name" on page 227	EP_SchemaName	The schema used for automatically creating the event recovery table.
"Enable Secure Network Connection" on page 227	SncMode	Indicates whether secure network connection mode is used.
"Event recovery data source (JNDI) name" on page 227	EP_DataSource_JNDIName	The JNDI name of the data source configured for event recovery.
"Event recovery table name" on page 228	EP_TableName	The name of the event recovery table.
Retry limit for failed events	FailedEventRetryLimit	The number of times the adapter attempts to redeliver an event before marking the event as failed
"Folder for RFC trace files" on page 228	RfcTracePath	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Gateway host" on page 229	GatewayHost	The host name of the SAP gateway.
"Gateway service" on page 229	GatewayService	The identifier of the gateway on the gateway host that carries out the RFC services.
"Host name" on page 230	ApplicationServerHost	Specifies the IP address or the name of the application server host that the adapter logs on to.
"Ignore IDoc packet errors" on page 230	IgnoreIDocPacketErrors	Determines what the adapter does when it encounters an error while processing the IDoc packet.
"Language code" on page 230	Language code	Specifies the Language code in which the adapter logs on to SAP.

Table 148. Activation specification properties for ALE inbound processing (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
"Logon group name" on page 231	Group	An identifier of the name of the group of application server instances that have been defined in transaction SMLG and linked together for logon load balancing.
"Maximum number of retries in case of system connection failure" on page 231	retryLimit	Specifies the number of times the adapter tries to restart the event listeners.
"Message server host" on page 232	MessageServerHost	Specifies the name of the host on which the message server is running.
"Number of listeners" on page 232	NumberOfListeners	Specifies the number of event listeners that are to be started.
"Partner character set" on page 232	PartnerCharset	Specifies PartnerCharset encoding.
"Password" on page 233	Password	The password of the user account of the adapter on the SAP application server.
"Password used to connect to event data source" on page 233	EP_Password	The user password for connecting to the database.
"RFC program ID" on page 233	RfcProgramID	The remote function call identifier under which the adapter registers in the SAP gateway.
"RFC trace level" on page 234	RfcTraceLevel	Specifies the global trace level.
"RFC trace on" on page 234	RfcTraceOn	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP system ID" on page 235	SAPSystemID	Specifies the system ID of the SAP system for which logon load balancing is allowed.
"Secure Network Connection library path" on page 235	SncLib	Specifies the path to the library that provides the secure network connection service.
"Secure Network Connection name" on page 235	SncMyname	Specifies the name of the secure network connection.
"Secure Network Connection partner" on page 236	SncPartnername	Specifies the name of the secure network connection partner.
"Secure Network Connection security level" on page 236	SncQop	Specifies the level of security for the secure network connection.
"System number" on page 236	SystemNumber	The system number of the SAP application server.
"Time between retries in case of system connection failure (milliseconds)" on page 237	retryInterval	Specifies the time interval between attempts to restart the event listeners.
"User name" on page 237	userName	The user account for the adapter on the SAP server.
"User name used to connect to event data source" on page 237	EP_UserName	The user name for connecting to the database.
"X509 certificate" on page 238	X509cert	Specifies the X509 certificate to be used as the logon ticket.

ALE failure code

The value entered determines how the adapter updates the SAP failure status code after the ALE module has retrieved an IDoc object for event processing.

Table 149. ALE failure code details

Required	Yes if AleUpdateStatus is set to `True`; no otherwise
Possible values	`68` `58`
Default	No default value.
Property type	Integer
Usage	Set a value for this property only if you set the value for AleUpdateStatus to `True`. Specify a value `68` for this property to cause the adapter to update the SAP failure status code after the ALE module has retrieved an IDoc object for event processing. SAP converts this value to 40 (Application Document not created in receiving system). When you set the AleUpdateStatus property to `True`, the adapter updates a standard SAP status code after the adapter retrieves an IDoc object for event processing. An IDoc that is not successfully sent to the endpoint is considered a failure. You use the ALE failure code property to specify the code used to signify this failure.
Globalized	No
Bidi supported	No

ALE failure text

The text that displays in the event that an IDoc is not successfully sent to the endpoint.

Table 150. ALE failure text details

Required	Yes if AleUpdateStatus is set to `True`; no otherwise.
Default	No default value.
Property type	String
Usage	Use this property only if you set the AleUpdateStatus property to `True`. The length of the text string cannot exceed 70 characters. When you set the AleUpdateStatus property to `True`, the adapter updates a standard SAP status code after the adapter retrieves an IDoc object for event processing. An IDoc that is not successfully sent to the endpoint is considered a failure. You use the ALE failure text property to specify the descriptive text used to signify this failure.
Example	`ALE Dispatch Failed`
Globalized	Yes
Bidi supported	No

ALE selective update

Specifies which IDoc Type and MessageType combinations are to be updated.

Table 151. ALE selective update details

Required	No
Default	No default value

Table 151. ALE selective update details (continued)

Property type	String
Usage	You can set values for this property only if AleUpdateStatus has been set to `True`. When you set the AleUpdateStatus property to `True`, the adapter updates a standard SAP status code after the adapter retrieves an IDoc object for event processing. You use the ALE selective update property to specify which IDoc Type and MessageType combinations are to be updated. The syntax for this property is: `IDocType: MessageType [;IDocType: MessageType [;...]]` where a slash (/) delimiter separates each IDoc Type and MessageType, and a semicolon (;) delimiter separates entries in a set.
Example	The following example illustrates two sets. In the example, MATMAS03 and DEBMAS03 are the IDocs, and MATMAS and DEBMAS are the message types: `MATMAS03/MATMAS;DEBMAS03/DEBMAS`
Globalized	No
Bidi supported	No

ALE status message code

This property specifies the message code to use when the adapter posts the ALEAUD01 IDoc with message type ALEAUD.

Table 152. ALE status message code details

Required	No
Possible values	For list of available codes, refer to the SAP table TEDS1.
Default	No default value.
Property type	String
Usage	• You can set a value for this property only if AleUpdateStatus has been set to `True`. • You must configure this message code in the receiving partner profile on SAP.
Globalized	No
Bidi supported	No

ALE success code

ALE success code for the successful posting of an IDoc.

Table 153. ALE success code details

Required	Yes if AleUpdateStatus is set to `True`; no otherwise
Possible values	`52` `53`
Default	No default value.
Property type	Integer

Table 153. ALE success code details (continued)

Usage	Use this property only if you set the AleUpdateStatus property to `True`. When you set the AleUpdateStatus property to `True`, the adapter updates a standard SAP status code after the adapter retrieves an IDoc object for event processing. You use the ALE success code property to specify the code for IDoc posted as 53. After the IDoc is sent to the endpoint, the IDoc status remains as 03 (IDoc posted to port) in SAP. After posting the IDoc, the adapter posts the audit IDoc with the current IDoc number and status as 53. SAP converts the current IDoc status to 41 (Application Document Created in Receiving System).
Globalized	No
Bidi supported	No

ALE success text

Indicates the text that displays when an application document is posted successfully.

Table 154. ALE success text details

Required	Yes if AleUpdateStatus is set to `True`; no otherwise.
Default	No default value.
Property type	String
Usage	Use this property only if you set the AleUpdateStatus property to `True`. The length of the text string cannot exceed 70 characters. When you set the AleUpdateStatus property to `True`, the adapter updates a standard SAP status code after the adapter retrieves an IDoc object for event processing. You use the ALE success text property to specify the descriptive text used to signify Application Document Posted.
Example	`ALE Dispatch OK`
Globalized	Yes
Bidi supported	No

ALE update status

This property specifies whether an audit trail is required for all message types.

Table 155. ALE update status details

Required	Yes
Possible values	`True` `False`
Default	`False`
Property type	Boolean

Table 155. ALE update status details (continued)

Usage	Set this property to `True` if you want the adapter to update a standard SAP status code after the ALE module has retrieved an IDoc object for event processing. If you set this value to `True`, you must also set following properties: • AleFailureCode • AleSuccessCode • AleFailureText • AleSuccessText.
Globalized	No
Bidi supported	No

Assured once-only delivery

This property specifies whether to provide assured once-only delivery for inbound events.

Table 156. Assured once-only delivery details

Required	No
Default	`False`
Property type	Boolean
Usage	When this property is set to `True`, the adapter provides assured once event delivery. This means that each event will be delivered **once and only once**. A value of `False` does not provide assured once event delivery, but provides better performance. When this property is set to `True`, the adapter attempts to store transaction (XID) information in the event store. If it is set to `False`, the adapter does not attempt to store the information. This property is used only if the export component is transactional. If the export component is not transactional, no transaction can be used, regardless of the value of this property.
Globalized	No
Bidi supported	No

Auto create event table

Determines if the event table is created automatically.

Table 157. Auto create event table details

Required	Yes, if **Assured once-only event delivery** is set to `True`, No otherwise.
Possible values	`True` `False`
Default	`True`
Property type	Boolean
Usage	This property indicates whether the adapter should create the event recovery table automatically if it does not already exist. In the administrative console, this property is listed as ″EP_CreateTable″. If you specify a value of `True` to automatically create the table, you must specify information about the event table (such as the event recovery table name). The value provided in the Event recovery table name property is used to create the table.

Table 157. Auto create event table details (continued)

Globalized	No
Bidi supported	No

Client

This property is the client number of the SAP system to which the adapter connects.

Table 158. Client details

Required	Yes
Possible values	You can enter a range of values from `000` to `999`.
Default	`100`
Property type	Integer
Usage	When an application attempts to log on to the SAP server, the SAP server requires that the application have a Client number associated with it. The Client property value identifies the client (the adapter) that is attempting to log onto the SAP server.
Globalized	No
Bidi supported	No

Codepage number

The numeric identifier of the code page.

Table 159. Codepage number details

Required	No
Possible values	You can enter a range of values from `0000` to `9999`. For a full listing of languages and associated codepage numbers supported by SAP, access SAP Note 7360.
Default	The default value for this property is conditionally determined by the value set for the **Language code** property.
Property type	Integer
Usage	The value assigned to the Codepage number defines the code page to be used and has a one-to-one relationship with the value set for the Language code property. The Codepage number establishes a connection to the appropriate language. Each language code value has a codepage number value associated with it. For example, the language code for English, is EN. If you selected EN (English) as your language code, the codepage number is automatically set to the numeric value associated with EN (English). The SAP code page number for EN (English) is `1100`.
Example	If **Language code** is set to `JA` (Japanese), **Codepage number** is set to `8000`.
Globalized	No
Bidi supported	No

Database schema name

This property is the schema used for automatically creating the event recovery table.

Note: In the administrative console, this property is listed as ″EP_SchemaName″.

Table 160. Database schema name details

Required	No
Default	No default value.
Property type	String
Usage	Specifies the schema name for the database used by the adapters event persistence feature.
Example	`ALE_SCHEMA`
Globalized	Yes
Bidi supported	No

Enable Secure Network Connection

This property indicates whether secure network connection mode is enabled.

Table 161. Enable Secure Network Connection details

Required	No
Possible values	`0` (off) `1` (on)
Default	`0`
Property type	String
Usage	Set the value to 1 (on) if you want to use secure network connection. If you set this value to 1, you must also set following properties: • “Secure Network Connection library path” on page 235 • “Secure Network Connection name” on page 235 • “Secure Network Connection partner” on page 236 • “Secure Network Connection security level” on page 236.
Globalized	No
Bidi supported	No

Event recovery data source (JNDI) name

This property is the JNDI name of the data source configured for event recovery.

Note: In the administrative console, this property is listed as ″EP_DataSource_JNDIName″.

Table 162. Event recovery data source (JNDI) name details

Required	Yes
Default	No default value.
Property type	String
Usage	Used in event recovery processing. The data source must be created in WebSphere Process Server. The adapter utilizes data source for *persisting* the event state.
Example	`jdbc/DB2`
Globalized	No
Bidi supported	No

Event recovery table name

This property is the name of the event recovery table.

Note: In the administrative console, this property is listed as ″EP_TableName″.

Table 163. Event recovery table name details

Required	Yes
Default	No default value.
Property type	String
Usage	Used in event recovery processing. Consult database documentation for information on naming conventions. It is recommended that a separate event recovery table is configured for each endpoint. The same data source can be used to hold all of the event recovery tables.
Example	`EVENT_TABLE`
Globalized	No
Bidi supported	No

Retry limit for failed events (FailedEventRetryLimit)

This property specifies the number of times that the adapter attempts to redeliver an event before marking the event as failed.

Table 164. Retry limit for failed events details

Required	No
Possible values	Integers
Default	`5`
Property type	Integer
Usage	Use this property to control how many times the adapter tries to send an event before marking it as failed. It accepts the following values: **Default** If this property is not set, the adapter tries five additional times before marking the event as failed. **0** The adapter tries to deliver the event an infinite number of times. When the property is set to `0`, the event remains in the event store and the event is never marked as failed. **> 0** For integers greater than zero, the adapter retries the specified number of times before marking the event as failed. **< 0** For negative integers, the adapter does not retry failed events.
Globalized	No
Bidi supported	No

Folder for RFC trace files

This property sets the fully qualified local path to the folder in which to write RFC trace files.

Table 165. Folder for RFC trace files details

Required	No
Default	No default value
Property type	String
Usage	Identifies the fully qualified local path into which RFC trace files are written. If **RFC trace on** is set to False (not selected), you are not permitted to set a value in the **Folder for RFC trace files** property.
Example	`c:\temp\rfcTraceDir`
Globalized	Yes
Bidi supported	No

Gateway host

This property is the Gateway host name. Enter either the IP address or the name of the Gateway host. Consult with your SAP administrator for information on the Gateway host name.

Table 166. Gateway host details

Required	Yes
Default	No default value
Property type	String
Usage	This property is the host name of the SAP gateway. The gateway enables communication between work processes on the SAP system and external programs. The host identified is used as the gateway for the resource adapter. Maximum length of 20 characters. If the computer name is longer than 20 characters, define a symbolic name in the THOSTS table.
Globalized	No
Bidi supported	No

Gateway service

This property is the identifier of the gateway on the gateway host that carries out the RFC services.

Table 167. Gateway service details

Required	Yes
Default	`sapgw00`
Property type	String
Usage	These services enable communication between work processes on the SAP server and external programs. The service typically has the format of `sapgw00`, where 00 is the SAP system number. Maximum of 20 characters.
Globalized	No
Bidi supported	No

Host name

Specifies the IP address or the name of the application server host that the adapter logs on to.

Table 168. Host name details

Required	Yes (when load balancing is not used).
Default	No default value
Property type	String
Usage	When the adapter is configured to run without load balancing, this property specifies the IP address or the name of the application server that the adapter logs on to.
Example	`sapServer`
Globalized	No
Bidi supported	No

Ignore IDoc packet errors

Determines whether or not IDoc packet errors are to be ignored.

Table 169. Ignore IDOC packet errors details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	If the adapter encounters an error while processing the IDoc packet, it can behave in two different ways. • When this property is set to `False`, the adapter stops processing further IDocs in that packet and reports an error to the SAP system. • When this property is set to `True`, the adapter logs an error and continues processing the rest of the IDocs in that packet. The status of the transaction is marked as `INPROGRESS`. The adapter log would display the IDoc numbers that failed and you need to resubmit those individual IDocs separately. You need to manually maintain these records in the event recovery table. This property is not used for single IDocs and for non-split IDoc packets.
Globalized	No
Bidi supported	No

Language code

This property specifies the Language code in which the adapter logs on.

Table 170. Language code details

Required	Yes
Possible values	For a full listing of languages and associated codepage numbers supported by SAP, access SAP Note 7360.
Default	The default value for the Language code property is based on the system locale.
Property type	String

Table 170. Language code details (continued)

Usage	Each of the supported languages is preceded by a 2 character language code. The language itself is displayed in parentheses. The language codes that display in the list represent the SAP default set of 41 languages for non Unicode systems plus Arabic. The value you select determines the value of the **Codepage number** property. If you manually enter a language code, you do not need to enter the language in parentheses.
Example	If the system locale is English, the value for this property is `EN (English)`.
Globalized	No
Bidi supported	No

Logon group name

This property is an identifier for the name of the group of application server instances that have been defined in transaction SMLG and linked together for logon load balancing.

Table 171. Logon group details

Required	Yes (if load balancing is used)
Possible values	Consult SAP documentation for information on creating Logon groups and on calling transaction SMLG.
Default	No default value
Property type	String
Usage	When the adapter is configured for load balancing, this property represents the name of the group of application server instances that have been defined in transaction SMLG and linked together for logon load balancing. Logon load balancing allows for the dynamic distribution of logon connections to application server instances. Maximum of 20 characters. On most SAP systems, the SPACE logon group is reserved by SAP.
Globalized	No
Bidi supported	No

Maximum number of retries in case of system connection failure

This property specifies the number of times the adapter tries to restart the event listeners.

Table 172. Maximum number of retries in case of system failure details

Required	Yes
Default	`0`
Property type	Integer

Table 172. Maximum number of retries in case of system failure details (continued)

Usage	When the adapter encounters an error related to the inbound connection (if the SAP application is down for example), this property specifies the number of times the adapter tries to restart the event listeners. A value of 0 indicates an infinite number of retries. **Note:** Configure the **Time between retries in case of system connection failure (milliseconds)** appropriately when retrying infinitely. For each retry attempt, the adapter waits based on the time interval specified in the **Time between retries in case of system connection failure (milliseconds)**. **Note:** If all the retry attempts fail, the adapter logs relevant messages and CEI events and stops attempting to recover the event listener. If you reach this point, you may need to restart the application manually.
Globalized	No
Bidi supported	No

Message server host

This property specifies the name of the host on which the message server is running.

Table 173. Message server host details

Required	Yes (if load balancing is used)
Default	No default value
Property type	String
Usage	This property specifies the name of the host that will inform all the servers (instances) belonging to this SAP system of the existence of the other servers to be used for load balancing. The message server host contains the information about load balancing for RFC clients so that an RFC client can be directed to an appropriate application server.
Example	`SAPERP05`
Globalized	No
Bidi supported	No

Number of listeners

This property specifies the number of listeners that are started by an event.

Table 174. Number of listeners details

Required	No
Default	1
Property type	Integer
Usage	For event sequencing, this property should be set to 1. To improve adapter performance, you can increase the number of listeners.
Globalized	No
Bidi supported	No

Partner character set

This property specifies the partner character set encoding.

Table 175. Partner character set details

Required	No
Default	UTF-8
Property type	String
Usage	When an encoding is specified, it is used; otherwise the default encoding is used.
Globalized	No
Bidi supported	No

Password

This property is the password of the user account of the adapter on the SAP application server.

Table 176. Password details

Required	Yes
Default	No default value
Property type	String
Usage	The restrictions on the password depend on the version of SAP Web Application Server. • For SAP Web Application Server version 6.40 or earlier, the password: – Must be uppercase – Must be 8 characters in length • For versions of SAP Web Application Server later than 6.40, the password: – Is not case-sensitive – Can be up to 40 characters in length
Globalized	No
Bidi supported	Yes

Password used to connect to event data source

This property is the user password for connecting to the database.

Note: In the administrative console, this property is listed as ″EP_Password″.

Table 177. Password to connect to event data source details

Required	Yes
Default	No default value.
Property type	String
Usage	This property specifies the password used by event persistence processing to obtain the database connection from the data source.
Globalized	Yes
Bidi supported	No

RFC program ID

This property is the program identifier under which the adapter registers in the SAP gateway.

Table 178. RFC program ID details

Required	Yes
Possible values	Use the SAP transaction SM59 (Display and Maintain RFC Destinations) to see a list of available RFC program IDs.
Default	No default value.
Property type	String
Usage	The adapter registers with the gateway so that listener threads can process events from RFC-enabled functions. This value must match the program ID registered in the SAP application. The maximum length is 64 characters.
Globalized	No
Bidi supported	No

RFC trace level

This property specifies the global trace level.

Table 179. RFC trace level details

Required	No
Possible values	`1` - This is the default RFC trace level. When specified, SAP JCo Java API logging occurs. `3` - When specified, SAP JCo JNI API logging occurs. `5` - When specified, error diagnostic logging occurs.
Default	1
Property type	Integer
Usage	If **RFC trace on** is set to `False` (not selected), you cannot set a value in the **RFC trace level** property.
Globalized	No
Bidi supported	No

RFC trace on

This property specifies whether to generate a text file detailing the RFC activity for each event listener.

Table 180. RFC trace on details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	A value of `True` activates tracing, which generates a text file. This file is created in the directory in which the adapter process was started. The file has a prefix of rfx and a file type of trc (for example, rfc03912_02220.trc). Use these text files in a development environment only, because the files can grow rapidly. If **RFC trace on** is set to `False` (not selected), you cannot set values in the **Folder for RFC trace files** or **RFC trace level** properties.

Table 180. RFC trace on details (continued)

Example	Examples of the information in the file are RfcCall FUNCTION BAPI_CUSTOMER_GETLIST, followed by the information for the parameters in the interface, or RFC Info rfctable, followed by the data from one of the interface tables. The trace file is created in the directory where the adapter process has been started. The trace file has a .trc file extension and the file name will start with the letters rfc followed by a unique identifier. For example, `rfc03912_02220.trc`.
Globalized	No
Bidi supported	No

SAP system ID

This property specifies the system ID of the SAP system for which logon load balancing is allowed.

Table 181. SAP system ID details

Required	Yes (when load balancing is used)
Default	No default value
Property type	String
Usage	Value must be three characters
Example	`DYL`
Globalized	No
Bidi supported	No

Secure Network Connection library path

This property specifies the path to the library that provides the secure network connection service.

Table 182. Secure Network Connection library path details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify the path to the library that provides the service.
Example	`/WINDOWS/system32/gssapi32.dll`
Globalized	No
Bidi supported	No

Secure Network Connection name

This property specifies the name of the secure network connection.

Table 183. Secure Network Connection name details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String

Table 183. Secure Network Connection name details (continued)

Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a name for the connection.
Example	DOMAINNAME/USERNAME
Globalized	No
Bidi supported	No

Secure Network Connection partner

This property specifies the name of the secure network connection partner.

Table 184. Secure Network Connection partner details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a name for the connection partner.
Example	CN=sap00.saperpdev, OU=Adapter, O=IBM, C=US
Globalized	No
Bidi supported	No

Secure Network Connection security level

This property specifies the level of security for the secure network connection.

Table 185. Secure Network Connection security level details

Required	Yes, if SncMode is set to 1; no otherwise.
Possible values	1 (Authentication only) 2 (Integrity protection) 3 (Privacy protection) 8 (Use the value from snc/data_protection/use on the application server) 9 (Use the value from snc/data_protection/max on the application server)
Default	3 (Privacy protection)
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a value to indicate the level of security for the connection.
Globalized	No
Bidi supported	No

System number

This property is the system number of the SAP application server.

Table 186. System number details

Required	Yes
Possible values	You can enter a range of values from 00 to 99.

Table 186. System number details (continued)

Default	00
Property type	Integer
Usage	The system number further identifies the Gateway service.
Globalized	No
Bidi supported	No

Time between retries in case of system connection failure (milliseconds)

This property specifies the time interval between attempts to restart the event listeners.

Table 187. Time between retries in case of system connection failure details

Required	Yes
Default	60000
Unit of measure	Milliseconds
Property type	Integer
Usage	When the adapter encounters an error related to the inbound connection, this property specifies the time interval the adapter waits in between attempts to restart the event listeners.
Globalized	No
Bidi supported	No

User name

This property is the user account for the adapter on the SAP server.

Table 188. User name details

Required	Yes
Default	No default value
Property type	String
Usage	Maximum length of 12 characters. The user name is not case sensitive. It is recommended that you set up a CPIC user account in the SAP application and that you give this account the necessary privileges to manipulate the data required by the business objects supported by the adapter. For example, if the adapter must perform certain SAP business transactions, the adapter's account in the SAP application must have the permissions set to allow it to perform these transactions.
Example	SapUser
Globalized	Yes
Bidi supported	Yes

User name used to connect to event data source

This property is the user name for connecting to the database.

Note: In the administrative console, this property is listed as ″EP_UserName″.

Table 189. User name to connect to event data source details

Required	Yes
Default	No default value.
Property type	String
Usage	User name used by event persistence for getting the database connection from the data source. Consult database documentation for information on naming conventions.
Globalized	Yes
Bidi supported	No

X509 certificate

This property specifies the X509 certificate to be used as the logon ticket.

Table 190. X509 certificate details

Required	No.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), you can provide a value for the X509 certificate.
Globalized	No
Bidi supported	No

Activation specification properties for Advanced event processing

Activation specification properties are properties that hold the inbound event processing configuration information for a message endpoint.

Activation specification properties are used during endpoint activation to notify the adapter of eligible event listeners. During inbound processing, the adapter uses these event listeners to receive events before forwarding them to the endpoint.

You set the activation specification properties using the J2C Bean wizard and can change them using the Rational Application Developer for WebSphere Software Assembly Editor, or after deployment through the WebSphere Application Server administrative console.

The following table lists the activation specification properties for Advanced event inbound processing. A complete description of each property is provided in the sections that follow the table. For information on how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 163.

Table 191. Activation specification properties for Advanced event processing

Property name		
In enterprise service wizard	**In administrative console**	**Purpose**
"Adapter instance for event filtering (AdapterInstanceEventFilter)" on page 240	AdapterInstanceEventFilter	Identifier that determines whether this adapter instance processes specific events in the event store

Table 191. Activation specification properties for Advanced event processing (continued)

Property name		
In enterprise service wizard	**In administrative console**	**Purpose**
"Assured once-only delivery" on page 241	AssuredOnceDelivery	Specifies whether to provide assured-once delivery for inbound events.
"Client" on page 242	Client	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 242	Codepage	Indicates the numeric identifier of the code page.
"Enable Secure Network Connection" on page 243	SncMode	Indicates whether secure network connection mode is used.
"Delivery type (DeliveryType)" on page 243	DeliveryType	Determines the order in which events are delivered by the adapter to the export
"Event types to process (EventTypeFilter)" on page 243	EventTypeFilter	A delimited list of event types that indicates to the adapter which events it should deliver
Retry limit for failed events	FailedEventRetryLimit	The number of times the adapter attempts to redeliver an event before marking the event as failed
"Folder for RFC trace files" on page 244	RfcTracePath	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Gateway host" on page 245	GatewayHost	The host name of the SAP gateway.
"Gateway service" on page 245	GatewayService	The identifier of the gateway on the gateway host that carries out the RFC services.
"Host name" on page 245	ApplicationServerHost	Specifies the IP address or the name of the application server host that the adapter logs on to.
"Language code" on page 246	Language code	Specifies the Language code in which the adapter logs on to SAP.
"Logon group name" on page 246	Group	An identifier of the name of the group of application server instances that have been defined in transaction SMLG and linked together for logon load balancing.
"Maximum number of events collected during each poll" on page 247	PollQuantity	The number of events that the adapter delivers to the export during each poll period
"Maximum number of retries in case of system connection failure" on page 247	RetryLimit	The number of times the adapter tries to reestablish an inbound connection after an error.
"Message server host" on page 247	MessageServerHost	Specifies the name of the host on which the message server is running.
"Partner character set" on page 248	PartnerCharset	Specifies PartnerCharset encoding.
"Password" on page 248	Password	The password of the user account of the adapter on the SAP application server.
Retry EIS connection on startup	RetryConnectionOnStartup	Controls whether the adapter retries the connection to the SAP server if it cannot connect at startup
"RFC trace level" on page 249	RfcTraceLevel	Specifies the global trace level.

Table 191. Activation specification properties for Advanced event processing (continued)

Property name		Purpose
In enterprise service wizard	**In administrative console**	
"RFC trace on" on page 249	RfcTraceOn	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP system ID" on page 250	SAPSystemID	Specifies the system ID of the SAP system for which logon load balancing is allowed.
"Secure Network Connection library path" on page 250	SncLib	Specifies the path to the library that provides the secure network connection service.
"Secure Network Connection name" on page 251	SncMyname	Specifies the name of the secure network connection.
"Secure Network Connection partner" on page 251	SncPartnername	Specifies the name of the secure network connection partner.
"Secure Network Connection security level" on page 251	SncQop	Specifies the level of security for the secure network connection.
"Stop the adapter when an error is encountered while polling (StopPollingOnError)" on page 252	StopPollingOnError	Specifies whether the adapter stops polling for events when it encounters an error during polling
"System number" on page 252	SystemNumber	The system number of the SAP application server.
"Time between polling for events (milliseconds)" on page 252	PollPeriod	The length of time that the adapter waits between polling periods
"Time between retries in case of system connection failure (milliseconds)" on page 253	RetryInterval	The length of time that the adapter waits between attempts to establish a new connection after an error during inbound operations
"User name" on page 253	userName	The user account for the adapter on the SAP server.
"X509 certificate" on page 253	X509cert	Specifies the X509 certificate to be used as the logon ticket.

Adapter instance for event filtering (AdapterInstanceEventFilter)

This property controls whether this adapter instance processes specific events in the event store.

Table 192. Adapter instance for event filtering details

Required	No
Default	`null`
Property type	String

Table 192. Adapter instance for event filtering details (continued)

Usage	This property helps you migrate from WebSphere Business Integration Adapter for mySAP to WebSphere Adapter for SAP Software. WebSphere Business Integration Adapter for mySAP allows you to perform load balancing on high-volume event types by allowing multiple adapter instances to process events of the same type. When load balancing is not required, a single adapter instance processes all events of a given type. This property is to enable seamless migration for WBIA customers to JCA for customers who are currently taking advantage of the connectorID filtering. WebSphere Adapter for SAP Software typically does not require load balancing in this way, but supports it so that you can migrate without modifying the database triggers or other mechanisms that write events to the event store. The AdapterInstanceEventFilter property corresponds to the ConnectorID property of the WebSphere Business Integration Adapter for mySAP. To use this feature, the database triggers or other mechanisms that create events in the event store must assign the appropriate value to the ConnectorId column. Table 193 shows the interaction between the AdapterInstanceEventFilter property and the value in the ConnectorId column in the event store. If the EventTypeFilter and AdapterInstanceEventFilter properties are both set, the adapter processes only events that meet both criteria. That is, it processes only those events whose type is specified in the EventTypeFilter property and whose ConnectorId column matches the AdapterInstanceEventFilter property.
Example	See Table 193.
Globalized	Yes
Bidi supported	Yes

Table 193. Interaction of the AdapterInstanceEventFilter property with the ConnectorId column in the event store

AdapterInstanceEventFilter property	**ConnectorId column of an event**	**Result**
`null`	`null`	The adapter processes the event
`null`	`Instance1`	The adapter processes the event, because the ConnectorId column is not checked
`Instance1`	`Instance1`	The adapter processes the event
`Instance1`	`Instance2`	The adapter does not process the event, because the instance IDs do not match
`Instance1`	`null`	The adapter does not process the event, because the instance IDs do not match

Assured once-only delivery

This property specifies whether to provide assured once-only delivery for inbound events.

Table 194. Assured once-only delivery details

Required	Yes
Default	`True`
Property type	Boolean

Table 194. Assured once-only delivery details (continued)

Usage	When this property is set to `True`, the adapter provides assured once event delivery. This means that each event will be delivered **once and only once**. A value of `False` does not provide assured once event delivery, but provides better performance. When this property is set to `True`, the adapter attempts to store transaction (XID) information in the event store. If it is set to `False`, the adapter does not attempt to store the information. This property is used only if the export component is transactional. If the export component is not transactional, no transaction can be used, regardless of the value of this property.
Globalized	No
Bidi supported	No

Client

This property is the client number of the SAP system to which the adapter connects.

Table 195. Client details

Required	Yes
Possible values	You can enter a range of values from `000` to `999`.
Default	`100`
Property type	Integer
Usage	When an application attempts to log on to the SAP server, the SAP server requires that the application have a Client number associated with it. The Client property value identifies the client (the adapter) that is attempting to log onto the SAP server.
Globalized	No
Bidi supported	No

Codepage number

The numeric identifier of the code page.

Table 196. Codepage number details

Required	No
Possible values	You can enter a range of values from `0000` to `9999`. For a full listing of languages and associated codepage numbers supported by SAP, access SAP Note 7360.
Default	The default value for this property is conditionally determined by the value set for the **Language code** property.
Property type	Integer
Usage	The value assigned to the Codepage number defines the code page to be used and has a one-to-one relationship with the value set for the Language code property. The Codepage number establishes a connection to the appropriate language. Each language code value has a codepage number value associated with it. For example, the language code for English, is EN. If you selected EN (English) as your language code, the codepage number is automatically set to the numeric value associated with EN (English). The SAP code page number for EN (English) is `1100`.
Example	If **Language code** is set to `JA` (Japanese), **Codepage number** is set to `8000`.

Table 196. Codepage number details (continued)

Globalized	No
Bidi supported	No

Delivery type (DeliveryType)

This property specifies the order in which events are delivered by the adapter to the export.

Table 197. Delivery type details

Required	No
Possible values	`ORDERED` `UNORDERED`
Default	`ORDERED`
Property type	String
Usage	The following values are supported: • `ORDERED`: The adapter delivers events to the export one at a time. • `UNORDERED`: The adapter delivers all events to the export at once.
Globalized	No
Bidi supported	No

Enable Secure Network Connection

This property indicates whether secure network connection mode is enabled.

Table 198. Enable Secure Network Connection details

Required	No
Possible values	`0` (off) `1` (on)
Default	`0`
Property type	String
Usage	Set the value to 1 (on) if you want to use secure network connection. If you set this value to 1, you must also set following properties: • "Secure Network Connection library path" on page 250 • "Secure Network Connection name" on page 251 • "Secure Network Connection partner" on page 251 • "Secure Network Connection security level" on page 251.
Globalized	No
Bidi supported	No

Event types to process (EventTypeFilter)

This property contains a delimited list of event types that indicates to the adapter which events it should deliver.

Table 199. Event types to process details

Required	No

Table 199. Event types to process details (continued)

Possible values	A comma-delimited (,) list of business object types
Default	`null`
Property type	String
Usage	Events are filtered by business object type. If the property is set, the adapter delivers only those events that are in the list. A value of `null` indicates that no filter will be applied and that all events will be delivered to the export.
Example	To receive only events relating to the Customer and Order business objects, specify this value: `Customer,Order` If the EventTypeFilter and AdapterInstanceEventFilter properties are both set, the adapter processes only events that meet both criteria. That is, it processes only those events whose type is specified in the EventTypeFilter property and whose ConnectorId column matches the AdapterInstanceEventFilter property.
Globalized	No
Bidi supported	No

Retry limit for failed events (FailedEventRetryLimit)

This property specifies the number of times that the adapter attempts to redeliver an event before marking the event as failed.

Table 200. Retry limit for failed events details

Required	No
Possible values	Integers
Default	`5`
Property type	Integer
Usage	Use this property to control how many times the adapter tries to send an event before marking it as failed. It accepts the following values: **Default** If this property is not set, the adapter tries five additional times before marking the event as failed. **0** The adapter tries to deliver the event an infinite number of times. When the property is set to `0`, the event remains in the event store and the event is never marked as failed. **> 0** For integers greater than zero, the adapter retries the specified number of times before marking the event as failed. **< 0** For negative integers, the adapter does not retry failed events.
Globalized	No
Bidi supported	No

Folder for RFC trace files

This property sets the fully qualified local path to the folder in which to write RFC trace files.

Table 201. Folder for RFC trace files details

Required	No
Default	No default value

Table 201. Folder for RFC trace files details (continued)

Property type	String
Usage	Identifies the fully qualified local path into which RFC trace files are written. If **RFC trace on** is set to False (not selected), you are not permitted to set a value in the **Folder for RFC trace files** property.
Example	`c:\temp\rfcTraceDir`
Globalized	Yes
Bidi supported	No

Gateway host

This property is the Gateway host name. Enter either the IP address or the name of the Gateway host. Consult with your SAP administrator for information on the Gateway host name.

Table 202. Gateway host details

Required	Yes
Default	No default value
Property type	String
Usage	This property is the host name of the SAP gateway. The gateway enables communication between work processes on the SAP system and external programs. The host identified is used as the gateway for the resource adapter. Maximum length of 20 characters. If the computer name is longer than 20 characters, define a symbolic name in the THOSTS table.
Globalized	No
Bidi supported	No

Gateway service

This property is the identifier of the gateway on the gateway host that carries out the RFC services.

Table 203. Gateway service details

Required	Yes
Default	`sapgw00`
Property type	String
Usage	These services enable communication between work processes on the SAP server and external programs. The service typically has the format of `sapgw00`, where 00 is the SAP system number. Maximum of 20 characters.
Globalized	No
Bidi supported	No

Host name

Specifies the IP address or the name of the application server host that the adapter logs on to.

Table 204. Host name details

Required	Yes (when load balancing is not used).
Default	No default value
Property type	String
Usage	When the adapter is configured to run without load balancing, this property specifies the IP address or the name of the application server that the adapter logs on to.
Example	`sapServer`
Globalized	No
Bidi supported	No

Language code

This property specifies the Language code in which the adapter logs on.

Table 205. Language code details

Required	Yes
Possible values	For a full listing of languages and associated codepage numbers supported by SAP, access SAP Note 7360.
Default	The default value for the Language code property is based on the system locale.
Property type	String
Usage	Each of the supported languages is preceded by a 2 character language code. The language itself is displayed in parentheses. The language codes that display in the list represent the SAP default set of 41 languages for non Unicode systems plus Arabic. The value you select determines the value of the **Codepage number** property. If you manually enter a language code, you do not need to enter the language in parentheses.
Example	If the system locale is English, the value for this property is `EN (English)`.
Globalized	No
Bidi supported	No

Logon group name

This property is an identifier for the name of the group of application server instances that have been defined in transaction SMLG and linked together for logon load balancing.

Table 206. Logon group details

Required	Yes (if load balancing is used)
Possible values	Consult SAP documentation for information on creating Logon groups and on calling transaction SMLG.
Default	No default value
Property type	String

Table 206. Logon group details (continued)

Usage	When the adapter is configured for load balancing, this property represents the name of the group of application server instances that have been defined in transaction SMLG and linked together for logon load balancing. Logon load balancing allows for the dynamic distribution of logon connections to application server instances. Maximum of 20 characters. On most SAP systems, the SPACE logon group is reserved by SAP.
Globalized	No
Bidi supported	No

Maximum number of events collected during each poll

This property specifies the number of events that the adapter delivers to the export during each poll period.

Table 207. Maximum number of events collected during each poll details

Required	Yes
Default	10
Property type	Integer
Usage	The value must be greater than 0
Globalized	No
Bidi supported	No

Maximum number of retries in case of system connection failure

This property specifies the number of times the adapter tries to reestablish an inbound connection.

Table 208. Maximum number of retries in case of system connection failure details

Required	No
Possible values	Positive integers
Default	0
Property type	Integer
Usage	Only positive values are valid. When the adapter encounters an error related to the inbound connection, this property specifies the number of times the adapter tries to restart the connection. A value of 0 indicates an infinite number of retries.
Globalized	No
Bidi supported	No

Message server host

This property specifies the name of the host on which the message server is running.

Table 209. Message server host details

Required	Yes (if load balancing is used)

Table 209. Message server host details (continued)

Default	No default value
Property type	String
Usage	This property specifies the name of the host that will inform all the servers (instances) belonging to this SAP system of the existence of the other servers to be used for load balancing. The message server host contains the information about load balancing for RFC clients so that an RFC client can be directed to an appropriate application server.
Example	`SAPERP05`
Globalized	No
Bidi supported	No

Partner character set

This property specifies the partner character set encoding.

Table 210. Partner character set details

Required	No
Default	UTF-8
Property type	String
Usage	When an encoding is specified, it is used; otherwise the default encoding is used.
Globalized	No
Bidi supported	No

Password

This property is the password of the user account of the adapter on the SAP application server.

Table 211. Password details

Required	Yes
Default	No default value
Property type	String
Usage	The restrictions on the password depend on the version of SAP Web Application Server. • For SAP Web Application Server version 6.40 or earlier, the password: – Must be uppercase – Must be 8 characters in length • For versions of SAP Web Application Server later than 6.40, the password: – Is not case-sensitive – Can be up to 40 characters in length
Globalized	No
Bidi supported	Yes

Retry EIS connection on startup (RetryConnectionOnStartup)

This property controls whether the adapter attempts to connect again to the SAP server if it cannot connect at startup.

Table 212. Retry EIS connection on startup details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	This property indicates whether the adapter should retry the connection to the SAP server if the connection cannot be made when the adapter is started: • Set the property to `False` when you want immediate feedback about whether the adapter can establish a connection to the SAP server, for example, when you are building and testing the application that receives events from the adapter. If the adapter cannot connect, the adapter writes log and trace information and stops. The administrative console shows the application status as `Stopped`. After you resolve the connection problem, start the adapter manually. • Set the property to `True` if you do not need immediate feedback about the connection. If the adapter cannot connect during startup, it writes log and trace information, and then attempts to reconnect, using the RetryInterval property to determine how frequently to retry and the value of the RetryLimit property to retry multiple times until that value is reached. The administrative console shows the application status as `Started`.
Globalized	No
Bidi supported	No

RFC trace level

This property specifies the global trace level.

Table 213. RFC trace level details

Required	No
Possible values	`1` - This is the default RFC trace level. When specified, SAP JCo Java API logging occurs. `3` - When specified, SAP JCo JNI API logging occurs. `5` - When specified, error diagnostic logging occurs.
Default	`1`
Property type	Integer
Usage	If **RFC trace on** is set to `False` (not selected), you cannot set a value in the **RFC trace level** property.
Globalized	No
Bidi supported	No

RFC trace on

This property specifies whether to generate a text file detailing the RFC activity for each event listener.

Table 214. RFC trace on details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean

Table 214. RFC trace on details (continued)

Usage	A value of `True` activates tracing, which generates a text file. This file is created in the directory in which the adapter process was started. The file has a prefix of rfx and a file type of trc (for example, rfc03912_02220.trc). Use these text files in a development environment only, because the files can grow rapidly. If **RFC trace on** is set to `False` (not selected), you cannot set values in the **Folder for RFC trace files** or **RFC trace level** properties.
Example	Examples of the information in the file are RfcCall FUNCTION BAPI_CUSTOMER_GETLIST, followed by the information for the parameters in the interface, or RFC Info rfctable, followed by the data from one of the interface tables. The trace file is created in the directory where the adapter process has been started. The trace file has a .trc file extension and the file name will start with the letters rfc followed by a unique identifier. For example, `rfc03912_02220.trc`.
Globalized	No
Bidi supported	No

SAP system ID

This property specifies the system ID of the SAP system for which logon load balancing is allowed.

Table 215. SAP system ID details

Required	Yes (when load balancing is used)
Default	No default value
Property type	String
Usage	Value must be three characters
Example	`DYL`
Globalized	No
Bidi supported	No

Secure Network Connection library path

This property specifies the path to the library that provides the secure network connection service.

Table 216. Secure Network Connection library path details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify the path to the library that provides the service.
Example	`/WINDOWS/system32/gssapi32.dll`
Globalized	No
Bidi supported	No

Secure Network Connection name

This property specifies the name of the secure network connection.

Table 217. Secure Network Connection name details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a name for the connection.
Example	`DOMAINNAME/USERNAME`
Globalized	No
Bidi supported	No

Secure Network Connection partner

This property specifies the name of the secure network connection partner.

Table 218. Secure Network Connection partner details

Required	Yes, if SncMode is set to 1; no otherwise.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a name for the connection partner.
Example	`CN=sap00.saperpdev, OU=Adapter, O=IBM, C=US`
Globalized	No
Bidi supported	No

Secure Network Connection security level

This property specifies the level of security for the secure network connection.

Table 219. Secure Network Connection security level details

Required	Yes, if SncMode is set to 1; no otherwise.
Possible values	`1` (Authentication only) `2` (Integrity protection) `3` (Privacy protection) `8` (Use the value from `snc/data_protection/use` on the application server) `9` (Use the value from `snc/data_protection/max` on the application server)
Default	`3` (Privacy protection)
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), specify a value to indicate the level of security for the connection.
Globalized	No
Bidi supported	No

Stop the adapter when an error is encountered while polling (StopPollingOnError)

This property specifies whether the adapter will stop polling for events when it encounters an error during polling.

Table 220. Stop the adapter when an error is encountered while polling details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	If this property is set to `True`, the adapter stops polling when it encounters an error. If this property is set to `False`, the adapter logs an exception when it encounters an error during polling and continues polling.
Globalized	No
Bidi supported	No

System number

This property is the system number of the SAP application server.

Table 221. System number details

Required	Yes
Possible values	You can enter a range of values from `00` to `99`.
Default	`00`
Property type	Integer
Usage	The system number further identifies the Gateway service.
Globalized	No
Bidi supported	No

Time between polling for events (milliseconds)

This property specifies the length of time that the adapter waits between polling periods.

Table 222. Time between polling for events (milliseconds)

Required	Yes
Possible values	Integers greater than or equal to `0`.
Default	`2000`
Unit of measure	Milliseconds
Property type	Integer
Usage	The time interval between polling events is established at a fixed rate, which means that if running the poll cycle is delayed for any reason (for example if a prior poll cycle takes longer than expected to complete) the next poll cycle will occur immediately to make up for the lost time caused by the delay.
Globalized	No

Table 222. Time between polling for events (milliseconds) (continued)

Bidi supported	No

Time between retries in case of system connection failure (milliseconds)

This property specifies the time interval between attempts to reestablish an inbound connection.

Table 223. Time between retries in case of system connection failure details

Required	Yes
Default	`60000`
Unit of measure	Milliseconds
Property type	Integer
Usage	When the adapter encounters an error related to the inbound connection, this property specifies the time interval the adapter waits in between attempts to reestablish an inbound connection.
Globalized	No
Bidi supported	No

User name

This property is the user account for the adapter on the SAP server.

Table 224. User name details

Required	Yes
Default	No default value
Property type	String
Usage	Maximum length of 12 characters. The user name is not case sensitive. It is recommended that you set up a CPIC user account in the SAP application and that you give this account the necessary privileges to manipulate the data required by the business objects supported by the adapter. For example, if the adapter must perform certain SAP business transactions, the adapter's account in the SAP application must have the permissions set to allow it to perform these transactions.
Example	`SapUser`
Globalized	Yes
Bidi supported	Yes

X509 certificate

This property specifies the X509 certificate to be used as the logon ticket.

Table 225. X509 certificate details

Required	No.
Default	No default value
Property type	String
Usage	If the SncMode property is set to 1 (indicating that you are using a secure network connection), you can provide a value for the X509 certificate.

Table 225. X509 certificate details (continued)

Globalized	No
Bidi supported	No

Globalization

WebSphere Adapter for SAP Software is a globalized application that can be used in multiple linguistic and cultural environments. Based on character set support and the locale of the host server, the adapter delivers message text in the appropriate language. The adapter supports bidirectional script data transformation between integration components.

Globalization and bidirectional transformation

The adapter is globalized to support single- and multi-byte character sets and deliver message text in the specified language. The adapter also performs bidirectional script data transformation, which refers to the task of processing data that contains both right-to-left (Hebrew or Arabic, for example) and left-to-right (a URL or file path, for example) semantic content within the same file.

Globalization

Globalized software applications are designed and developed for use within multiple linguistic and cultural environments rather than a single environment. WebSphere Adapters, Rational Application Developer for WebSphere Software, and WebSphere Application Server are written in Java. The Java runtime environment within the Java virtual machine (JVM) represents data in the Unicode character code set. Unicode contains encodings for characters in most known character code sets (both single- and multi-byte). Therefore, when data is transferred between these integration system components, there is no need for character conversion.

To log error and informational messages in the appropriate language and for the appropriate country or region, the adapter uses the locale of the system on which it is running.

Bidirectional script data transformation

Languages such as Arabic and Hebrew are written from right to left, yet they contain embedded segments of text that are written left to right, resulting in bidirectional script. When software applications handle bidirectional script data, standards are used to display and process it. Bidirectional script data transformation applies only to string type data. WebSphere Application Server uses the Windows standard format, but applications or file systems exchanging data with the server might use a different format. The adapter transforms bidirectional script data passed between the two systems so that it is accurately processed and displayed on both sides of a transaction. It transforms the script data by using a set of properties that defines the format of script data, as well as properties that identify content or metadata to which transformation applies.

Bidirectional data formats

Languages such as Arabic and Hebrew are written from right to left, yet they contain embedded segments of text that are written left to right, resulting in bidirectional script. When software applications handle bidirectional script, standards are used to display and process it. WebSphere Application Server use the

Windows standard format, but an enterprise information system exchanging data with WebSphere Application Server can use a different format. WebSphere Adapters transform bidirectional script data passed between the two systems so that it is accurately processed and displayed on both sides of a transaction.

Bidirectional format

WebSphere Application Server use the bidirectional format of ILYNN (implicit, left-to-right, on, off, nominal). This is the format used by Windows. If an enterprise information system uses a different format, the adapter converts the format prior to introducing the data to WebSphere Application Server.

Five attributes comprise bidirectional format. When you set bidirectional properties, you assign values for each of these attributes. The attributes and settings are listed in the following table.

Table 226. Bidirectional format attributes

Letter position	Purpose	Values	Description	Default setting
1	Order schema	I	Implicit (Logical)	I
		V	Visual	
2	Direction	L	Left-to-Right	L
		R	Right-to-Left	
		C	Contextual Left-to-Right	
		D	Contextual Right-to-Left	
3	Symmetric Swapping	Y	Symmetric swapping is on	Y
		N	Symmetric swapping is off	
4	Text Shaping	S	Text is shaped	N
		N	Text is not shaped (Nominal)	
		I	Initial shaping	
		M	Middle shaping	
		F	Final shaping	
		B	Isolated shaping	
5	Numeric Shaping	H	National (Hindi)	N
		C	Contextual shaping	
		N	Numbers are not shaped (Nominal)	

Bidirectional properties that identify data for transformation

To identify business data subject to transformation, set the BiDiContextEIS property. Do this by specifying values for each of the five bidirectional format attributes (listed in the table in the previous section) for the property. The BiDiContextEIS property can be set for the managed connection factory and the activation specification.

To identify event persistence data subject to transformation, set the BiDiFormatEP property. Do this by specifying values for each of the five bidirectional format

attributes (listed in the table in the previous section) for the property. The BiDiFormatEP property can be set for the activation specification.

To identify application-specific data for transformation, annotate the BiDiContextEIS property and the BiDiMetadata property within a business object. Do this by using the business object editor within Rational Application Developer for WebSphere Software to add the properties as application-specific elements of a business object.

Properties enabled for bidirectional data transformation

Bidirectional data transformation properties enforce the correct format of bidirectional script data exchanged between an application or file system and integration tools and runtime environments. Once these properties are set, bidirectional script data is correctly processed and displayed in Rational Application Developer for WebSphere Software and WebSphere Application Server

Enterprise service discovery connection properties

The following enterprise service discovery connection properties control bidirectional script data transformation.

- UserName
- Password

Managed connection factory properties

The following managed connection properties control bidirectional script data transformation.

- UserName
- Password

Activation specification properties

The following activation specification properties control bidirectional script data transformation.

- UserName
- Password

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department 2Z4A/SOM1
294 Route 100
Somers, NY 10589-0100
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: (c) (your company name) (year). Portions of

this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning:

Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A complete and current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

This product includes software developed by the Eclipse Project (http://www.eclipse.org).

Index

A

ABAP Debug property 178
ABAP handlers
  creation 47
  overview 46
activation specification properties
  list of 204, 219, 238
  setting in administrative console 129, 132
adapter
  project, create 67
Adapter for SAP Software
  administering 127
adapter log file
  configuring 137
  displaying 137
  truncating 138
Adapter packaging 6
Advanced event processing (AEP) interface
  ABAP handlers 46, 47
  batch programs 62
  business objects 53
  business workflows 64
  Call Transaction Recorder wizard 48
  change pointers 65
  custom triggers 60
  inbound processing
    configuring business objects 124
    overview 49
    selecting business objects 123
  managing 133
  outbound processing
    configuring business objects 107
    overview 46
    selecting business objects 106
  overview 3, 4, 45
  transport files 59
  WebSphere BI Station tool 133
Advanced event processing business objects
  application-specific information 154
  business-object-level metadata 154
  metadata 154
  naming conventions 163
  operation-level metadata 156
  operations 159
  parameters 155
  property-level metadata 155
ALE business objects
  application-specific information 151
  business-object-level metadata 151
  IDoc status codes 35, 42
  metadata 151
  operation-level metadata 153
  operations 157, 158
  parameters 152
  property-level metadata 152
ALE failure code property 35, 42, 222
ALE failure text property 222
ALE interface
  business objects
    metadata 151
    naming conventions 161
ALE interface *(continued)*
  business objects *(continued)*
    structure 36
  inbound processing
    configuring business objects 119
    creating data source 57
    discovering IDocs from file 118
    discovering IDocs from system 116
    error handling 31, 40
    overview 29, 39
    selecting business objects 116, 121
  outbound processing
    configuring business objects 91
    discovering IDocs from file 90
    discovering IDocs from system 89
    overview 29, 38
    selecting business objects 88, 95
  overview 3, 4, 27, 37
ALE pass-through IDoc interface
  business objects
    structure 43
ALE selective update property 222
ALE status message code property 223
ALE success code property 35, 42, 223
ALE success text property 35, 42, 224
ALE update status property 35, 42, 224
ALEAUD IDoc 35, 42
application-specific information
  Advanced event processing business objects 154
  ALE business objects 151
  BAPI business objects 149
  Query interface for SAP Software business objects 153
archive table 135
archived events
  deleting 136
  displaying 135
  resubmitting 135
Assured once-only delivery property 22, 32, 42, 206, 225, 241
authentication
  description 12
  J2C Bean wizard 12
  run time 13
authentication alias 13
Auto create event table property
  description 206, 225
  prerequisite 57

B

BAPI business objects
  business-object-level metadata 149
  naming conventions 160
  nested 24
  operation-level metadata 150
  operations 156
  parameters 150
  property-level metadata 150
  result set 27
  simple 22
  work units 25

BAPI inbound interface
configuring business objects 113
overview 4
selecting business objects 112
BAPI interface
configuring result set business objects 84
configuring simple business objects 71
inbound processing 19, 20
overview 3, 17, 26
BAPI outbound interface
configuring work unit business objects 78
outbound processing 18
selecting business objects 70
BAPI result set interface
selecting business objects 83
BAPI result set outbound interface
outbound processing 26
BAPI result sets
business object structure 27
overview 3, 26
BAPI work unit interface
overview 24
selecting business objects 77
BAPI work unit outbound interface
outbound processing 24
BAPI work units
business-object structure 25
overview 3
rollback mechanism 25
batch programs 62
BI Station tool 133
BQPROC field 22, 32, 42
BQTOTAL field 22, 32, 42
business object information 149
business objects
Advanced event processing interface
business object-level metadata 154
metadata 154
naming conventions 163
operation-level metadata 156
operations 159
property-level metadata 155
structure 53
ALE interface
IDoc status codes 35, 42
metadata 151
naming conventions 161
operations 157, 158
structure 36
ALE pass-through IDoc interface
structure 43
BAPI
result set 27
simple 22
work unit 25
BAPI interface
business-object-level metadata 149
metadata 149
naming conventions 160
operation-level metadata 150
operations 156
property-level metadata 150
overview 6
Query interface for SAP Software
business-object-level metadata 153
metadata 153
naming conventions 162
business objects *(continued)*
Query interface for SAP Software *(continued)*
operations 158
overview 44
property-level metadata 153
structure 44
business workflows 64
business-object-level metadata
Advanced event processing business objects 154
ALE business objects 151
BAPI business objects 149
Query interface for SAP Software business objects 153

C

Call Transaction Recorder wizard 48
change pointers 65
Client property 168, 179, 195, 207, 226, 242
clustered environment
deploying in 14
description 14
inbound processes 15
outbound processes 15
Codepage number property 168, 179, 196, 207, 226, 242
compatibility matrix 2
confidential data, disguising 11
confidential tracing 11
configuring
logging 141
tracing 141
connection properties, J2C Bean wizard 68
connector project 67
control record, IDoc 36
Create operation 158, 159
current events queue 133
custom properties
activation specification 129, 132
managed connection factory 128, 131
resource adapter 127, 130
Custom retrieve function name property 188
custom triggers 60

D

data record, IDoc 36
data source
creating 57
JNDI name 57
overview 21, 31, 41
troubleshooting 58
database connection, testing 58
database drivers, location 58
Database schema name property 208, 226
debugging
self-help resources 147
XAResourceNotAvailableException exception 146
definition file, IDoc 58
Delete operation 158, 159
deployment
options 13
distribution model 56

E

embedded adapter
activation specification properties, setting 129

embedded adapter *(continued)*
considerations for using 14
description 13
managed connection factory properties, setting 128
resource adapter properties, setting 127
enableHASupport property 15
endpoints, multiple 30, 39
EP_CreateTable property
description 21, 31, 41, 206, 225
prerequisite for using 57
EP_DataSource_JNDIName property 209, 227
EP_Password property 214, 233
EP_SchemaName property 208, 226
EP_TableName property 209, 228
EP_UserName property 219, 237
error handling, event 31, 40
Error in ASSIGN statement in the program SAPLSDTX 145
ErrorCode, setting 144
ErrorConfiguration, setting 144
ErrorDetail, setting 144
ErrorParameter, setting 144
errors
JCo Server could not unmarshall tables 145
out-of-memory 145
event detection 50
event processing
parsed IDoc packets 32
unparsed IDoc packets 34
event queue
current 133
future 134
event recovery 29, 39
Event recovery data source (JNDI) name property 209, 227
Event recovery table name property 209, 228
event recovery table, ALE 31, 41
event recovery table, BAPI 21
event restriction 52
event triggers 51
EVNTDATA field 22, 32, 42
EVNTID field 21, 32, 41
EVNTSTAT field 21, 32, 42
exceptions
Error in ASSIGN statement in the program SAPLSDTX 145
XAResourceNotAvailableException 146
Execute operation 157
Exists operation 158
export file 8
external dependencies, adding 68

F

FFDC (first-failure data capture) 146
files
IDoc definition 58
SystemOut.log log file 143
trace.log trace file 143
first-failure data capture (FFDC) 146
Folders for RFC trace files 169, 181, 196, 210, 228, 244
Function name property 188
future events queue 134

G

gateway connections, monitoring 139
Gateway host property 181, 210, 229, 245
Gateway service property 182, 211, 229, 245

H

hardware and software requirements 2
hardware requirements 2
high-availability environment
deploying in 14
description 14
inbound processes 15
outbound processes 15
Host name property 169, 182, 197, 211, 230, 245

I

IDoc definition file 58
IDoc packets
parsed 32
unparsed 34
IDocs
control record 36
data record 36
definition 27, 37
inbound processing 29, 39
outbound processing 29, 38
status codes 35, 42
Ignore errors in BAPI return property 190
Ignore IDoc packet errors property 230
import file 8
inbound configuration properties 191
inbound processing
Advanced event processing interface 49
ALE 29, 39
BAPI 19
BAPI interface 20
overview 2
interaction specification properties
Custom retrieve function name 188
Function name 188
Ignore errors in BAPI return 190
Maximum number of hits for the discovery 190
Select the queue name 190
interaction specification property
description 187

J

J2C Bean wizard
authentication in 12
overview 8
properties, connection 165, 192
setting connection properties 68
J2C local transactions 5
JAR file, adding external 68
Java 2 security 13
JCo Server could not unmarshall tables error 145
JDBC provider 57

L

Language code property 170, 182, 197, 211, 230, 246
local transactions 5
Log Analyzer 142
Log and Trace Analyzer, support for 141
log and trace files 141

Log file output location property 170, 197
log files
  changing file name 143
  disabling 142
  enabling 142
  level of detail 142
  location 143
logging
  configuring properties with administrative console 142
Logging level property 170, 198
logging options 137
logical system 56
Logon group name property 212, 231, 246

M

managed (J2C) connection factory properties
  list of 176
  setting in administrative console 128, 131
matrix, compatibility 2
Maximum number of events collected during each poll property 247
Maximum number of events collected property 247
Maximum number of hits for the discovery property 190
Maximum number of retries in case of system connection failure property 212, 231, 247
Maximum number of retries property 212, 231, 247
memory-related errors 145
Message server host property 183, 213, 232, 247
metadata
  business-object level
    Advanced event processing 154
    ALE 151
    BAPI 149
    Query interface for SAP Software 153
  operation-level
    Advanced event processing 156
    ALE 153
    BAPI 150
  property-object level
    Advanced event processing 155
    ALE 152
    BAPI 150
    Query interface for SAP Software 153

N

naming conventions
  Advanced event processing business objects 163
  ALE business objects 161
  BAPI business objects 160
  Query interface for SAP Software business objects 162
nested BAPI 24
Number of listeners property 213, 232

O

operation-level metadata
  Advanced event processing business objects 156
  ALE business objects 153
  BAPI business objects 150
operations, supported
  Advanced event processing inbound 159
  Advanced event processing outbound 159
  ALE inbound 158
  ALE outbound 157
operations, supported *(continued)*
  BAPI interface 156
  Query interface for SAP Software 158
out-of-memory errors 145
outbound configuration properties 163
outbound processing 5, 6
  Advanced event processing 46
  ALE 29, 38
  BAPI interface 18
  BAPI result set interface 26
  BAPI work unit interface 24
  overview 2
  Query interface for SAP Software 44

P

package files for adapters 142
Partner character set property 183, 213, 232, 248
partner profile 56
Password property 171, 183, 198, 214, 233, 248
Password to connect to event data source property 214, 233
problem determination
  self-help resources 147
  XAResourceNotAvailableException exception 146
program ID, RFC 55
properties
  activation specification 129, 132
    list of 204, 219, 238
  configuration properties
    inbound 191
    outbound 163
  external service connection 165, 192
  inbound configuration 191
  managed (J2C) connection factory 128, 131
    list of 176
  outbound configuration 163
  resource adapter 127, 130
    list of 174, 201
property-level metadata
  Advanced event processing business objects 155
  ALE business objects 152
  BAPI business objects 150
  Query interface for SAP Software business objects 153

Q

qRFC protocol 27, 37
Query interface for SAP Software
  business objects 44
  configuring business objects 102
  Error in ASSIGN statement in the program SAPLSDTX 145
  outbound processing 44
  overview 3, 43
  selecting business objects 100
Query interface for SAP Software business objects
  business-object-level metadata 153
  naming conventions 162
  operations 158
  parameters 153
  property-level metadata 153
  structure 44
querying data in SAP tables 44

R

RAR (resource adapter archive) file
  versions of 5
receiver port 55
requirements, hardware and software 2
resource adapter archive (RAR) file
  versions of 5
resource adapter properties
  list of 174, 201
  setting in administrative console 127, 130
result sets, BAPI
  business object structure 27
  overview 26
Retrieve operation 159
RetrieveAll operation 158
Retry Interval property 31, 40
Retry Limit property 31, 40
RFC program ID
  description 215, 233
  registering 55
RFC trace level 172, 184, 199, 215, 234, 249
RFC trace on 172, 184, 199, 215, 234, 249
RFC trace path folder 169, 181, 196, 210, 228, 244
runtime environment
  authentication in 13

S

SAP gateway connections, monitoring 139
SAP Interface name property 173, 200
SAP system ID property 185, 216, 235, 250
SAP tables 44
sapjco.jar file 68
Secure Network Connection library path property 185, 216, 235, 250
Secure Network Connection name property 185, 217, 235, 251
Secure Network Connection partner property 186, 217, 236, 251
Secure Network Connection security level property 186, 217, 236, 251
security
  disguising sensitive data 11
security, Java 2 13
Select the queue name 190
self-help resources 147
sensitive data, disguising 11
setting connection properties 68
simple BAPI
  business object structure 22
  description 17
SncLib property 185, 216, 235, 250
SncMode property 180, 208, 227, 243
SncMyname property 185, 217, 235, 251
SncPartnername property 186, 217, 236, 251
SncQop property 186, 217, 236, 251
software dependencies, adding external 68
software requirements 2
stand-alone adapter
  activation specification properties, setting 132
  considerations for using 14
  description 13
  managed connection factory properties, setting 131
  resource adapter properties, setting 130
status codes, IDocs 35, 42
support
  overview 141
support *(continued)*
  self-help resources 147
System number property 173, 186, 201, 218, 236, 252
SystemOut.log file 143

T

technotes 2, 147
TID (transaction identifier) 27, 37
Time between retries in case of system connection failure 218, 237, 253
Time between retries property 218, 237, 253
trace files
  changing file name 143
  disabling 142
  enabling 142
  level of detail 142
  location 143
trace.log file 143
tracing
  configuring properties with administrative console 142
transaction identifier (TID) 27, 37
transport files 59
tRFC protocol 21, 27, 32, 37, 41
triggers, event 51
troubleshooting
  data source creation 58
  overview 141
  self-help resources 147
  XAResourceNotAvailableException exception 146

U

Update operation 158, 159
User name property 174, 187, 201, 218, 237, 253
User name used to connect to event data source property 219, 237

W

WebSphere Adapter for SAP Software
  ALE interface 35
  BAPI interface 22
  BAPI result set 27
  BAPI work unit 25
  Java data binding 23, 24, 25, 27
  Java data bindings 22, 35
  nested BAPI 24
  overview 1
  SAP interfaces 17
  simple BAPI 23
  structure for a BAPI result set 27
  structure for a BAPI work unit 25
  structure for nested BAPI 24
  structure for simple BAPI 23
WebSphere Extended Deployment 15
work units, BAPI
  business object structure 25
  overview 24
wrapper, business object
  ALE 36, 43
  BAPI 23
  BAPI work unit 25

X

X509 certificate property 187, 219, 238, 253
XAResourceNotAvailableException 146
XID field 22, 32, 42

Printed in USA