IBM WebSphere Adapters
Version 7 Release 5 Fix Pack 2 (7.5.0.2)

IBM WebSphere Adapter for SAP Software User Guide Version 7 Release 5 Fix Pack 2 (7.5.0.2)

IBM WebSphere Adapters
Version 7 Release 5 Fix Pack 2 (7.5.0.2)

IBM WebSphere Adapter for SAP Software User Guide Version 7 Release 5 Fix Pack 2 (7.5.0.2)

Note
Before using this information and the product it supports, read the information in "Notices" on page 309.

May 2012

This edition applies to version 7, Release 5, Fix Pack 2 (7.5.0.2) 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

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 Platform, Enterprise Edition (JEE) 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. Therefore, the application component used by the promotions department needs to understand the low-level API and the data structures of the SAP function to call the function. Information technology resources and time would be needed to create the linkage between the application component and the SAP function.

With 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 does not 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 website.

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: http://www-947.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_Adapters_Family.
- Technotes for WebSphere Adapters provide workaround 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 invoked 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 can 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 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.

Monitoring inbound events

The adapter supports monitoring inbound events from the SAP server, in addition to the other events you are monitoring using Business Monitor or WebSphere Business Events.

Monitoring inbound events using IBM Business Monitor:

You can use Rational Application Developer for WebSphere Software and the adapter to send inbound events to WebSphere Application Server Common Event Infrastructure (CEI), where they are accessible to Business Monitor.

When you select the option to monitor inbound events in the Rational Application Developer for WebSphere Software J2C Bean wizard, the required artifacts are generated to monitor inbound events. These artifacts include the message-driven J2C bean, as well as the interface, Service Bean, interceptor class, helper class, and the event schemas that are required to create a monitor model. You can then deploy the resulting adapter inbound event monitoring application containing the message-driven bean (the adapter application) to a server, either a Business Monitor server or a remote server. The message-driven bean invokes the stateless session bean that makes the events accessible to the client. More importantly, it also listens for events coming in from the SAP server (inbound events) and uses the interceptor to set up the intercepted inbound events as Common Base Events (CBE). Then, it posts these Common Base Events into a designated Java Message Service (JMS) destination - Common Event Infrastructure queue, where they are accessible to Business Monitor for further processing.

Important: Inbound event monitoring is available to your application only if you have Business Monitor installed in your environment. For more information about installing Business Monitor, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v8r0mx/topic/com.ibm.wbpm.mon.imuc.doc/inst/intro.html. For more

information about the software requirements and configuration, see http://www-304.ibm.com/support/docview.wss?uid=swg27008414.

For enabling inbound event monitoring function, perform the following tasks:

1. Configure the SAP system (see "Configuration on the SAP system" on page 57)
2. Launch the J2C Bean wizard (see "Launching the J2C Bean wizard" on page 70)
3. Configure the connector dependencies (see "Configuring the connector dependencies" on page 70)
4. Set the connection properties in J2C Bean wizard (see "Setting connection properties for the J2C Bean wizard" on page 71)
5. Configuring the module for inbound processing (see "Configuring the module for inbound processing" on page 129)

Related References

For a sample on enabling inbound event monitoring for Business Monitor, see http://publib.boulder.ibm.com/infocenter/radhelp/v8r5/topic/com.ibm.j2c.doc/topics/tcreatinginboundapps.html.

For information about how to disable the event monitor function, see http://publib.boulder.ibm.com/infocenter/radhelp/v8r5/topic/com.ibm.j2c.doc/topics/tdisablingwbe.html.

For a sample end-to-end scenario on publishing events to the Business Monitor, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v8r0mx/topic/com.ibm.wbpm.mon.doc/scen/eis.html.

Monitoring inbound events using WebSphere Business Events:

You can use Rational Application Developer for WebSphere Software and the adapter to send inbound events to WebSphere Application Server JMS topic, where they are accessible to WebSphere Business Events.

Note: You cannot create a JMS connection to the remote server when the same connection factory name is duplicated. For more information, see http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.pmc.doc/tasks/tjn0033_.html

When you select the option to monitor inbound events in the Rational Application Developer for WebSphere Software J2C Bean wizard, the required artifacts are generated to monitor inbound events. These artifacts include the message-driven J2C bean, as well as the interface, Service Bean, interceptor class, helper class, and the event schemas that are required to create a monitor model. You can then deploy the resulting adapter inbound event monitoring application containing the message-driven bean (the adapter application) to a server. The message-driven bean invokes the stateless session bean that makes the events accessible to the client. More importantly, it also listens for events coming in from the SAP server (inbound events) and uses the interceptor to set up the intercepted inbound events as Common Base Events (CBE). Then, it posts these Common Base Events into a designated Java Message Service (JMS) destination - JMS topic, where they are accessible to WebSphere Business Events for further processing.

Important: Inbound event monitoring is available to your application only if you have WebSphere Business Events installed in your environment. For information about installing WebSphere Business Events, see http://publib.boulder.ibm.com/

infocenter/wbevents/v6r2m1/index.jsp?topic=/com.ibm.wbe.install.doc/doc/install.html. The WebSphere Business Events works with WebSphere Application Server version 6.1; it is not supported in WebSphere Application Server version 7.0. For more information about the software requirements and configuration, see http://www.ibm.com/software/integration/wbe/requirements/.

For enabling inbound event monitoring function, perform the following tasks:

1. Configure the SAP system (see "Configuration on the SAP system" on page 57)
2. Launch the J2C Bean wizard (see "Launching the J2C Bean wizard" on page 70)
3. Configure the connector dependencies (see "Configuring the connector dependencies" on page 70)
4. Set the connection properties in J2C Bean wizard (see "Setting connection properties for the J2C Bean wizard" on page 71)
5. Configuring the module for inbound processing (see "Configuring the module for inbound processing" on page 129)
6. Generate the eventBOTypeMapping.xml and eventBOTypeMapping.xsd files from the generated inbound session bean. The eventMapping file provides the mapping between the event and the business object schema that the WebSphere Business Event requires for monitoring the event. To generate the eventBOTypeMapping.xml and eventBOTypeMapping.xsd files:
   a. Right-click your session bean.
   b. Select **Source** > **Generate Event Mapping**.

   The EventMapping files get generated in the same folder as your business object schema files.

Related Reference

For integrating WebSphere Business Events with WebSphere Application Adapters, see http://publib.boulder.ibm.com/infocenter/wbevents/v6r2m1/topic/com.ibm.wbe.integrating.doc/doc/integratingusingwebsphereadapters.html.

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.

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.

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

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.

Business objects

The business object is a structure or a container to exchange data between application components and the adapter. The data can represent either a business entity, such as an invoice or an employee record, or unstructured text.

For outbound processing, the application component uses business objects to send data to SAP or to obtain data (through the adapter) from SAP. In other words, the application component 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.

During adapter configuration, you can optionally choose to generate a business graph. In version 6.0.2, each top-level business object is contained in a business graph, which includes a verb that an application can use to specify additional information about the operation to be performed. Beginning version 7.0, business graphs are optional; they are required only when you are adding business objects to a module created with an earlier version. If business graphs exist, they are processed, but the verb is ignored.

Authentication using connection specification properties

WebSphere Adapter for SAP Software uses connection properties either through Managed Connection Factory properties or a Java Authentication and Authorization Services (JAAS) alias. If you want to change the connection properties used for authentication with either one of these authentication methods, you can change the connection properties through the IBM Business Process Manager administrative console and restart the Java EE application or change the JAAS security settings.

In addition to the methods explained, the connection parameters can also be specified through the ConnectionSpec properties. The ConnectionSpec properties are used by an application component to pass connection-related properties.

You can specify the relevant ConnectionSpec properties for the outbound request. If you specify both ConnectionSpec properties and Managed Connection Factory properties during run time, the adapter uses the values specified in the ConnectionSpec properties to create a connection and ignores the values in the Managed Connection Factory properties.

The following table lists the properties available to enable Secure Network Connection (SNC):

Table 1. Secure Network connection properties

Authentication mechanism	Properties
Secure Network Connection (SNC)	sncMyname, sncPartnername, sncQop, sncMode, sncLib

Refer to “Enable Secure Network Connection” on page 228, for more information.

Note: The SSO token / X509 authentication mechanism can also be accomplished by setting the username to "$X509CERT$" (for X509 certificate based authentication) and "$MYSAPSSO2$" (for SSO token based authentication) and setting the password to a base 64 encoded string, which comprises of a SSO token / X509 certificate.

To configure the adapter to create an SAP server connection, see “Passing the connection parameters dynamically” on page 76.

Service Bean

The business data exchanged between the client application and the resource adapter is represented as Service Bean. The metadata describing the business data

is defined as business objects and represented as the XSD schemas. The Service Bean is generated from these XSDs and is the realization of the business objects.

A Service Bean 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 Service Bean by using the J2C Bean wizard, launched from Rational Application Developer for WebSphere Software connector tools. The wizard connects to the SAP Software, discovers data structures in the EIS, and generates Service Bean 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.

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 using standard text editors and scripts 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 website http://www.ibm.com/able/ for more information about the commitment that IBM has to accessibility.

Internet Protocol, Version 6 (IPv6)

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.

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, ensure that the first eight characters of the adapter ID property are unique for each instance so that you can correlate the 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 the log and trace information to a particular instance of an adapter. For example, when 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. If you set the adapter ID properties of two instances to `Instance01` and `Instance02`, you will not be able to 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. It prevents 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, see the “Adapter ID (AdapterID)” on page 217 property.

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 must possess a thorough understanding of the business integration concepts, the capabilities, and requirements of the integration development tools and the runtime environment you will use, and the SAP 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 the 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

You can configure the adapter to prevent sensitive or confidential data, in the log and trace files, from being viewed by users 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 might 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 such situations, the adapter lets you specify whether 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 219 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 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 SAP server to process outbound requests and inbound events.

Authentication in the wizard

The J2C Bean wizard prompts for the 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 WebSphere 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 is completed.

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. You can configure the adapter to get your user information, through 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 must 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 affect the type of deployment option you choose.

The following are the deployment options:

- 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.
- **With module for use by single application**: With the adapter files embedded in the module, you can deploy the module to any application server. Use an embedded adapter when you have a single module using the adapter or if multiple modules need to run different versions of the adapter. Using an embedded adapter enables you to upgrade the adapter in a single module without the risk of destabilizing other modules by changing their adapter version.
- **On server for use by multiple applications**: If you do not include the adapter files in a module, you must install them as a stand-alone adapter on each application server where you want to run the module. Use a stand-alone adapter when multiple modules can use the same version of the adapter and you want to administer the adapter in a central location. A stand-alone adapter can also reduce the resources required by running a single adapter instance for multiple modules.

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 affects how the adapter is used in the run time 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 to run 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. This provides the RAR file that you install following instructions supplied with WebSphere Application Server.

Considerations for embedding an adapter in the application

Consider the following items if you plan to embed 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 run time environments. *Class loader isolation* means that 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

Consider the following items if you plan to use 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.

Considerations while deploying adapters with different versions

When you install multiple adapters with different versions of CWYBS_AdapterFoundation.jar, and if a lower version of the CWYBS_AdapterFoundation.jar is loaded during runtime, the adapter will return the ResourceAdapterInternalException error message, due to a version conflict. For example, when you install Oracle E-Business Suite adapter version 7.0.0.3 and WebSphere Adapter for SAP Software version 7.5.0.2, the following error message is displayed "The version of CWYBS_AdapterFoundation.jar is not compatible with IBM WebSphere Adapter for SAP Software" as IBM WebSphere Adapter for SAP Software loads file:/C:/IBM/WebSphere/ProcServer7/profiles/ProcSrv01/installedConnectors/CWYOE_OracleEBS.rar/CWYBS_AdapterFoundation.jar with version 7.0.0.3. However, the base level of this jar required is version 7.5.0.2. To overcome this conflict, you must ensure that all adapters are at same version level. For further assistance, contact WebSphere Adapters Support for help.

There are occasions when you have to work with embedded adapters that do not need a client-server communication, standalone adapters that need a server connection, or a hybrid mix of adapter connections.

The following scenarios cover the different behaviors of AFC version conflict detection, when you are deploying two or more adapters and at least one of the adapter version is 7.5 or higher.

Deploying a standalone Adapter

1. Install WebSphere Adapter for Siebel Business Applications version 7.0.1.0 through the WebSphere Application Server administrative console.
2. Install WebSphere Adapter for SAP Software version 7.5.0.0 through the administrative console.
3. Create ActivationSpec for an ALE passthrough inbound operation.
4. Create an application in Rational Application Developer for WebSphere Software for a standalone ALE passthrough inbound operation.
5. Install and start the application through the administrative console.
6. Verify the error.

Note: An error message will be generated in the log/trace area of WebSphere Application Server, to indicate an AFC version conflict.

Deploying an embedded Adapter

1. Import a build of WebSphere Adapter for PeopleSoft Enterprise version 7.0.1.0, using a RAR file.
2. Create a Peoplesoft Inbound EMD operation.
3. Import a build of WebSphere Adapter for Oracle E-Business Suite version 7.5.0.0, using a RAR file.
4. Create an Oracle inbound EMD operation, in the same module where you have created the Peoplesoft Inbound EMD operation.

5. Deploy the module to WebSphere Application Server.
6. Check the trace.

At step 5, the deployment will fail. At step 6, you will get an internal error message due to the AFC version conflict.

Note: To avoid a name conflict between the business object generated by the two adapters, you may need to generate the artifacts into different folders.

Deploying a combination of standalone and embedded Adapters

1. Install WebSphere Adapter for Oracle E-Business Suite version 7.0.1.0 through theWebSphere Application Server administrative console.
2. Create an ActivationSpec for a Oracle inbound operation.
3. Create an application in Rational Application Developer for WebSphere Software for a Oracle inbound operation, for the standalone Adapter deployment.
4. Deploy the Oracle inbound application and trigger your inbound events.
5. Create an application in Rational Application Developer for WebSphere Software for a WebSphere Adapter for SAP Software version 7.5.0.0 inbound embedded Adapter deployment.
6. Deploy an SAP inbound application, and trigger your inbound events.

Note: You can resolve the AFC version conflict by using different class loaders for the standalone and embedded deployments. You can start both Oracle and SAP inbound applications successfully, and process Inbound events without exception.

For further assistance, visit http://www-947.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_Adapters_Family.

WebSphere Adapters in clustered environments

You can improve adapter performance and availability by deploying a module on a clustered server environment. Clusters are groups of servers that are managed together to balance workloads and to provide high availability and scalability.

The module you deployed is replicated across all servers in a cluster, regardless of whether you deploy the module using a stand-alone or an embedded adapter. The following IBM products support WebSphere Adapters in a clustered environment:

- WebSphere Application Server
- WebSphere Application Server Network Deployment
- WebSphere Extended Deployment

To deploy and configure WebSphere Adapter for SAP Software in a clustered environment, see: “Deploying the module in a clustered environment” on page 164. When you set up a server cluster, you create a Deployment Manager profile. The HAManager, a subcomponent of the Deployment Manager, notifies the Java Platform, Enterprise Edition (JEE) Connector Architecture (JCA) container to activate an adapter instance.

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 instance instead of a static Workload Manager. The dynamic Workload Manager instance 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 systems with different capacities and configurations to handle load variations evenly.

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 Platform, Enterprise Edition (JEE) 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.

Note: In the active-passive configuration mode of the adapters, the endpoint application of the passive adapter instance also listens to the events/messages even if the `enableHASupport` property is set to `True`. This is because the `alwaysactivateAllMDBs` property in the JMS activation specification is set to `True`. To stop the endpoint application of the passive adapter instance from listening to the events, you must set the `alwaysactivateAllMDBs` property value to `False`. For more information, see Disabling end point applications of the passive adapter .

If the value for the enableHASupport property is false, all adapter instances will listen for events on the Inbound cluster. Any number of SAP adapters can be made Active on a HA cluster – all in the active mode. When more than one adapter instance actively polls in a cluster set up, it serves as a load balancer. If one of the cluster instances fails, the other Active instances in the cluster will handle the events. If an IDoc fails, when SAP resubmits it, the other Active instances of the Adapter will handle the events.

Note: In clustered environments, when the adapter works in a Active-Active configuration, it provides both high availability and load balancing support. This functionality is useful in production environments where high performance is needed.

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

WebSphere Adapter for SAP Software supports outbound processing and inbound processing for simple BAPIs. In outbound processing, the 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 J2C bean 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.

The adapter supports the following data types in the import, export, and changing parameter list:

1. Elementary types
2. Complex data type
   a. Table types
      1) Line types
         a) Structure (example - BAPIRET2)
         b) Table type (example - TRTEXTS)
         c) Data Element (example - TRACKTEXT)
         d) Views (example - T001W_BIW)
      2) Predefined types (example - BIC_ADD_DATA_TT)
   b. Structure

The adapter supports the following data types in the table parameter list -

1. Tables with flat line structure (example - BAPIRET2)

Note: If you select the **Generate business objects in SAP XI standard** option in the Discovery Configuration window, the table type element will have a wrapper anonymous complex type, whose name is appended with a suffix "**_Item**".

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).

During configuration you can make a selection 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.

Wait on BAPI Commit

When using the BAPI interface to create data on SAP, the data will not be committed to the SAP Database until the BAPI_TRANSACTION_COMMIT BAPI is invoked explicitly. When creating a large amount of data using a BAPI, if you invoke the BAPI_TRANSACTION_COMMIT without setting the WAIT parameter, the Update process is initiated on SAP and exits immediately without waiting for the Update process to complete.

If you select **Wait until commit call to SAP database is completed and returned** during configuration, the adapter will wait until all time critical updates to SAP database are completed before invoking the commit call. You can set the value for the Wait on a BAPI Commit call to be either a non-space character, which is interpreted as 'True', or set it a space character, which is interpreted as 'False.' If you set the value to 'True', then the BAPI call will wait indefinitely until the update process is completed before exiting.

There is a performance impact when you use the WAIT parameter as processing is delayed till the database update is completed. The need for a Wait on a BAPI Commit should to be evaluated based on your requirement.

If it is not selected, the commit call is performed immediately regardless of whether all the execution within a transaction is complete or not.

Note: This functionality is supported for BAPI interface and BAPI work unit interface.

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 system, and if the scheduler in SAP is configured, then the call is repeated at regular intervals until it can be processed by the adapter. If the scheduler is not configured, you need to process it manually.

   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 the 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, 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 2. 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 the BAPI transactions are transferred.
EVNTSTAT	INTEGER	Event processing status. Possible values are: • 0 (Created) • 1 (Executed) • 3 (In Progress) • -1 (Rollback)

Table 2. 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 beans for the BAPI interface

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

A Java bean 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 beans 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 beans. This metadata contains BAPI-related information such as the operation of the Java bean, table parameters, transaction information, and dependent or grouped BAPIs.

You create Java beans 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 beans 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 bean 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`. For more information on naming conventions, refer to "Naming conventions for BAPI business objects" on page 200.

Note: If you select the option in the Discovery Configurationwindow, the table type element will have a wrapper anonymous complex type, whose name is appended with a suffix "**_Item**".

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.

Business objects for BAPI can also be generated without a wrapper. This is the recommended approach. If you are generating business objects without a wrapper, the execute operation is associated with each of the selected BAPIs by default. In the case of **Asynchronous Transactional RFC** or **Asynchronous Queued RFC** the transaction ID property is present in the top-level business object.

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.

The BAPI work unit interface

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.

Note: This object, which contains the results of the BAPI operation, is named according to the convention `Sap + Name of the structure`. For more information about naming conventions, refer to "Naming conventions for BAPI business objects" on page 200.

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.

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 about 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.

Note: This object, which contains the results of the BAPI operation, is named according to the convention `Sap + Name of the structure`. For more information about naming conventions, refer to "Naming conventions for BAPI business objects" on page 200.

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.

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. The SAP adapter supports the basic and extension type of IDocs.

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 CWYAP_SAPAdapter_Tx.rar and have provided a Transaction ID, the adapter uses it while posting the IDocs to the SAP system. If you have not provided a Transaction ID, the adapter will create one before posting. 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. The adapter uses the same Transaction ID for all the IDocs in the wrapper and posts all of them using one call.
4. The adapter throws an exception if the data record is empty. Refer to "Java bean structure for the ALE interface" on page 39 for more information on data records.
5. 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 configure 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.

If you have selected split IDocs and SAP is sending packet IDocs, the adapter delivers each IDoc inside the packet as an individual event to the end point.

During recovery, the SAP system has to resubmit the whole packet. The adapter only delivers IDocs which have not been delivered in previous attempts from the packet.

Note: The adapter is able to listen to and deliver events from multiple SAP systems using multiple activation specs.

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 3. differences between the ALE interface and the ALE pass-through IDoc interface

Interface	When to use	SplitIDoc = true	SplitIDoc = false	Parsed IDoc = true	Parsed IDoc = false
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 parsed for both control record and data record. The individual segments in the IDoc are read and parsed to convert into business objects.	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.

Table 3. differences between the ALE interface and the ALE pass-through IDoc interface (continued)

Interface	When to use	SplitIDoc = true	SplitIDoc = false	Parsed IDoc = true	Parsed IDoc = false
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.	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 relies on SAP JCo's retry handling to restart the JCo Server.
   - The adapter stops the server after the SAP JCo Server tries again for the specified RetryLimit value as specified in the activation specification.

   Note: If the retry interval does not behave as specified in the activation specification, download the latest SAP JCo 3.0.6.
3. If all the retry attempts fail, the adapter logs the relevant message and CEI events, stops message end points and stops trying to recover the ALE event listener.

   Note: You must restart the adapter or SCA application in this case.
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 enabled by 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 4. 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. During adapter configuration, you can specify 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 as `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 as 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:

- Success Code
- Failure Code
- Success Text
- Failure Text

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 5. 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 bean 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 contains a control record, a data record and a dummy key..

Control record

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

The control record can be generated from SAP field names or from SAP field descriptions. While configuring the properties for the control record you can specify if you want the control record generated from SAP field names or from SAP field descriptions. Check the check box to use SAP field names to generate attribute names if you want the control record generated from field names.

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. The data record business object is generated for a parsed IDoc. The data record business object contains all the segments of the IDoc. Each segment in turn has a child business object as shown below. The segment attributes can also be generated using SAP field names or field descriptions. You can use SAP field names to generate attribute names.

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. The IDoc data is of hexBinary type and represents the whole data record containing segments in binary content.

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. You can use dummy keys in the ALE inbound interface.

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. The Adapter supports basic and extension IDoc types.

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 supports multiple IDocs in the data stream. The data stream content can be of two types:
   a. Inline format without a delimiter - as supported in previous versions. In the inline format, multiple IDocs are differentiated using the header **EDIDC_40**
   b. Delimited content format - supported in the current version. In the delimited format, you can post multiple IDocs to the SAP system by using a delimiter in between, to achieve the standard length of 1063 characters as specified by the SAP adapter. You can use delimiters to mark IDoc boundaries and also mark Control and Data Records boundaries within an IDoc.

      When you insert a delimiter between Data Records, the adapter can then identify each Data Record and pad them to meet SAP specifications. The different uses of the delimiter are as follows:
      - Single IDoc:
      - Single IDoc with '\n' character to separate data records
      - Multiple IDocs
      - Multiple IDocs with the '\n' character separating the Data Records

Figure 1. Multiple IDocs with the '\n' character separating the Data Records

3. The adapter uses the IDoc wrapper business object to populate the appropriate RFC-enabled function call used by the ALE interface.
4. The adapter establishes an RFC connection to the ALE interface and passes the IDoc data to the SAP system. If you are using the CWYAP_SAPAdapter_Tx.rar and have provided a Transaction ID, adapter uses it while posting the IDoc to SAP. If you have not provided a Transaction ID, the adapter will create one before posting. 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. The adapter uses the same Transaction ID for all the IDocs in the wrapper and posts all of them using one call.
5. 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 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.

You can also choose to send the IDoc in Flat File format to the end point. You make this selection on the Configuration Parametersscreen using the **Send IDoc in flat file format** check box. This feature will only work if the adapter is configured to work with unparsed Control Records where the **Parse the IDoc Control Record** check box is disabled.

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.

If you have selected split IDocs and SAP is sending packet IDocs, the adapter delivers each IDoc inside the packet as an individual event to the end point. During recovery, the SAP system has to resubmit the whole packet. The adapter only delivers IDocs which have not been delivered in previous attempts from the packet.

Note that the adapter is able to listen to and deliver events from multiple SAP systems using multiple activation specs.

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 6. Differences between ALE interface and ALE pass-through interface

Interface	When to use	SplitIDoc = true	SplitIDoc = false	Parsed IDoc = true	Flat File format = 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.	not applicable
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.	Control record of the incoming IDoc is parsed. In case of the packet IDoc, only the first IDoc from the control record will be parsed.	The incoming IDoc would be sent as a HexBinary in Flat File format, where each record (control and data record) is separated with a delimiter.

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 relies on SAP JCo's retry handling to restart the JCo Server.
   - The adapter stops the server after the SAP JCo Server tries again for the specified RetryLimit value as specified in the activation specification.

   Note: If the retry interval does not behave as specified in the activation specification, download the latest SAP JCo 3.0.6.
3. If all the retry attempts fail, the adapter logs the relevant message and CEI events, stops message end points and stops trying to recover the ALE event listener.

   Note: You must restart the adapter or SCA application in this case.

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 enabled by 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 7. 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`.

Table 7. Event recovery table fields (continued)

Table field name	Type	Description
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 as 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:

- Success Code
- Failure Code
- Success Text
- Failure Text

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 8. 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 inbound operation using pass-through IDocs, you can choose to create a parsed Control Record (Inbound only) along with a HexBinary stream to hold both the Control Record and the Data Record. For the first case, the business object will contain a child Business Object representing the parsed Control Record, and a HexBinary field to hold both the Data Record and Control Record. The same business object format is used for inbound and outbound ALE pass-through IDoc processing. For outbound operations using pass-through IDocs, the business object contains an IDoc stream instead of a child business object.

The control record will be generated using field names, by default.

Parsed Control Record for Inbound

The business object contains a transaction ID, a queue name, stream data to hold the Data Record and Control Record, a child business object for the parsed Control Record 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.

To parse the IDoc control record, select the checkbox in the configuration screen during the J2C Bean wizard run.

When the **Parse the IDoc Control Record** checkbox is selected, a child business object is generated to hold the parsed Control Record.

The following figure illustrates a business object that represents a Generic IDoc when the checkbox is selected:

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.

The ASI for the delimiter and the IDoc version will also be added if the Flat File option is enabled.

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 run time. 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 the integrity of the database might get corrupted . 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 9. Interface parameters

Parameter	Description
OBJECT_KEY_IN	Should be no value.
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 the connection to the SAP system is lost or the application is stopped before the event is delivered to the endpoint, 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 10. 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_FUT_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 is 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_FUT_AEP with the event.
2. The /CWLD/ADD_TO_QUEUE_IN_FUT_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 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_FUT_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 configure authorization profiles and 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. Contact 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: Only the first task is required for BAPI inbound processing.

Procedure

1. Access levels required for the user name that is used to connect to the SAP system.:

   To run the SAP adapter smoothly, configure the following authorization profiles in the SAP system:

Table 11.

OBJECT	DESCRIPTION	AUTHORIZATION
B_ALE_RECV ALE/EDI:	Receiving IDocs via RFC	B_ALE_RC_ALL
S_CTS_ADMI	Administration Functions in the Change and Transport S	S_CTS_IMPALL
S_RFCACL	Authorization Check for RFC User (for example, Trusted System)	S_RFCACL_ALL
S_TCODE	Authorization Check for Transaction Start	S_TCD_ALL
S_RFC	Authorization check for RFC access	S_RFC_ALL

Table 11. (continued)

OBJECT	DESCRIPTION	AUTHORIZATION
S_TABU_DIS	Table Maintenance (via standard tools such as SM30)	S_TABU_ALL
S_IDOCCTRL	WFEDI: S_IDOCCTRL - General Access to IDoc Functions	S_IDCCTR_AL+
S_IDOCDEFT	WFEDI: S_IDOCDEFT - Access to IDoc Development	S_IDCDFT_ALL

To identify which authorizations are absolutely necessary, perform the following steps:

a. Open **TCode SM19** and use the Security Audit trace
b. Run the SAP Adapter
c. Refer to the system log **SM20** for authorization objects that are either accessed or denied

2. 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. Under the **MDMP and Unicode** tab, set the RFC destination as **Unicode** or **non-Unicode** by choosing the appropriate radio button.

      For error-free operation of the adapter while using multiple language settings, set the RFC destination as Unicode.
   h. Save your entry.
3. 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.
4. 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.
5. 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.

6. 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_7.5.0.1_xx>\transports`. Change the version according to your current installation.

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

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 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

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

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

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

Ensure that 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 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**. Click **Next**.
2. 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., and then click **Next**.
3. In the **Connector Import** window, accept the default project name in the **Connector project** field or type a different name.
4. In the **Target server** field, select the type of server where you want to deploy the module. The wizard creates the artifacts that are appropriate to that server.
5. Click **Next**. The Connector Settings window is displayed.

What to do next

Continue working in the J2C Bean wizard. The next step is to add dependent JAR files to the project.

Configuring the connector dependencies

As part of generating the service, you are prompted by the J2C Bean wizard to specify the location of the required sapjco3.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 sapjco3.jar file and the associated files for your operating system from your SAP administrator or from the SAP Web site. Obtain the CWYAP_SAPAdapterExt.jar from the adapter package. The files are listed in the following table.

Table 12. Connector dependency files required by SAP Software

Operating system	Files to be copied
Windows and i5/OS®	sapjco3.jar, sapidoc3.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®)	sapidoc3.jar, sapjco3.jar and any .so and .o files that come with the SAP JCo download from the SAP Web site

2. SAP JCo3 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. Add the CWYAP_SAPAdapterExt.jar from the SAP adapter package.
4. 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.

      Note: You are prompted for the location of the .dll files only if they are not already located in the Windows system path.
   b. Click **Next**.

Results

The `sapjco3.jar` file and associated files are now part of your project.

What to do next

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 information such 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 software dependency files (the sapjco3.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 Adapter Style window, 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) and click **Next**. The Discovery Configuration window is displayed.
2. Optional: Event monitoring is available to your application only if you have Business Monitor or WebSphere Business Events installed in your environment.
   - Use the following procedure, to generate and monitor common base events and manage these events with the Common Event Infrastructure (CEI) services using Business Monitor:
     a. Select the Enable Inbound Event Monitor check box and then click **Next**. The **Event and JMS configuration** window is displayed.
     b. In the **Event type** field, select WebSphere Business Monitor.
     c. In the **Queue connectionFactory JNDI name** field, accept the default value, `jms/cei/EventQueueConnectionFactory`.
     d. In the **Queue JNDI name** field, accept the default value, `jms/cei/EventQueue`.
     e. Click **Advanced** to add advanced properties.
     f. In the Remote JNDI provider configuration section, specify the naming provider URL host name and port name values in the **Naming provider URL Host** and **Naming provider URL port** fields to connect to the remote WebSphere Application Server from the wizard.
     g. In the Connection authentication configuration section, type the user name in the **User name** field to connect to the server from the wizard.
     h. In the Connection authentication configuration section, type the password in the **Password** field to connect to the server from the wizard.
     i. Click **Next**. The Discovery Configuration window is displayed.
   - Use the following procedure, to generate and monitor common base events and manage these events using WebSphere Business Events:
     a. Select the Enable Inbound Event Monitor check box and then click **Next**. The **Event and JMS configuration** window is displayed.
     b. In the **Event type** field, select WebSphere Business Events.
     c. In the **Topic connectionFactory JNDI name** field, accept the default value, `jms/WbeTopicConnectionfactory` .
     d. In the **Topic JNDI name** field, accept the default value, `jms/WBE/CbeListener`.
     e. Click **Advanced** to add advanced properties.
     f. The **Remote JNDI provider configuration** field is used to configure the remote topics.

        If the bus in the local cell has the same name as the bus in remote cell, the application always connect to the local cell. It does not use any of the provider endpoints specified on the connection factory, so the Remote Topic Configuration information that you enter is ignored. For more information about remote Topic Configuration, refer to http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.pmc.doc/tasks/tjn0033_.html .
     g. In the **Naming provider URL Host** field, specify the naming provider URL host name to connect to the remote WebSphere Application Server from the wizard.

h. In the **Naming provider URL port** field, specify the naming provider URL port name to connect to the remote WebSphere Application Server from the wizard.

i. In the Connection authentication configuration section, type the user name in the **User name** field to connect to the server from the wizard.

j. In the Connection authentication configuration section, type the password in the **Password** field to connect to the server from the wizard.

k. Click **Next**. The Discovery Configuration window is displayed.

3. 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`.

      Note: You will only get a listing of IDocs in the language selected. The error messages will be displayed in the specified language.

   e. Type the name and password you use to access the SAP server.

      The password is case-sensitive.

   f. Choose an interface from the following **SAP interface name** options:

      - Advanced Event Processing (AEP), ALE, ALE pass-through IDoc or BAPI for Inbound, or

Figure 2. SAP interfaces - inbound

      - Advanced Event Processing (AEP), ALE, ALE pass-through IDoc, BAPI, BAPI work unit, BAPI result set or Query interface for SAP Software for Outbound

Figure 3. SAP interfaces - outbound

      Note: This option is not available if you are modifying existing artifacts.

4. Optional: To set additional advanced properties, click **Advanced**.

5. To set RFC tracing properties, perform the following steps:

   a. Expand **SAP RFC trace configuration** and select **RFC trace on**.

   b. Select the **Write JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API trace logged into the broker trace file will have the string `[JCoAPI]`. All JCo API traces are logged at the `INFO` message log level.

   c. Select a tracing level from the **RFC trace level** list.

   d. Click **Browse** and select a location to which the RFC trace files will be saved.

Note: Enabling RFC tracing in turn enables CPIC tracing as well. CPIC tracing cannot be disabled individually.

Note: To avoid tracing files getting created in multiple locations, create an environment variable (for example RFC_TRACE_DIR) and set it to a valid folder.

If a value is set for the environment variable RFC_TRACE_DIR, all RFC will be generated in the folder set in the environment variable RFC_TRACE_DIR

All CPIC traces will be generated in the current working directory.

If the environment variable RFC_TRACE_DIR is not set, traces generated are saved under the folder `<RAD_ROOT>`.. This is the top-level folder of your IID installation. For example, `C:\IBM\IID7.5`

6. Optional: To enable bidirectional support for the adapter at run time:
   a. In the **Bidi properties** area, select **Bidi transformation**.
   b. Set the ordering schema, text direction, symmetric swapping, character shaping, and numeric shaping properties to control how bidirectional transformation is performed.
7. To configure Business Object name generated by the adapter for the BAPI Modules, expand **Business Object naming configuration** and select the required options from the following:
   a. Check the checkbox to have the adapter to generate all the Business Object names same as that of the SAP XI standard. This option is applicable only for BAPI outbound processing.
   b. By default, the **Enforce the same naming convention for business objects** checkbox is selected to have the adapter generate all the Business Object names without appending any hashcode. The hashcode is appended to the namespace of each Business Object instead of the name.
   c. Do not check the **Enforce the same naming convention for business objects** checkbox to have the adapter append the hashcode to the subsequent Business Object that has the same name. This is done to avoid duplication.

   For more information on naming conventions, refer to "Naming conventions for BAPI business objects" on page 200
8. Optional: To change the location of the log files for the wizard or the amount of information included in the logs, click **Specify the level of the logging desired**, and then provide the following information:
   a. In **Log file output location**, specify the location of the log file for the wizard.
   b. In **Logging level**, specify the severity of errors that you want logged.

      Note: This log information is for the wizard only; at run time, the adapter writes messages and trace information into the standard log and trace files for the server.
9. 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.

What to do next

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 the business object definitions and related artifacts.

Before you begin

The following table outlines the properties that are applicable for an interface:

Table 13.

Interface	SAP RFC Trace Configuration	Bidi properties	Business Objects naming convention	
			SAP PI naming convention	Enforce naming convention for business objects
BAPI	Yes	Yes	Yes	Yes
BAPI Work unit	Yes	Yes	No	Yes
BAPI Result Set	Yes	Yes	No	Yes
ALE	Yes	Yes	Yes	No
ALE Pass-through Idoc	Yes	Yes	No	No
Query Interface for SAP Software (QISS)	Yes	Yes	No	No
Advanced Event Process (AEP)	Yes	Yes	No	No

Authentication using connection specification properties

WebSphere Adapter for SAP Software uses connection properties either through Managed Connection Factory properties or a Java Authentication and Authorization Services (JAAS) alias. If you want to change the connection properties used for authentication with either one of these authentication methods, you can change the connection properties through the IBM Business Process Manager administrative console and restart the Java EE application or change the JAAS security settings.

In addition to the methods explained, the connection parameters can also be specified through the ConnectionSpec properties. The ConnectionSpec properties are used by an application component to pass connection-related properties.

You can specify the relevant ConnectionSpec properties for the outbound request. If you specify both ConnectionSpec properties and Managed Connection Factory properties during run time, the adapter uses the values specified in the ConnectionSpec properties to create a connection and ignores the values in the Managed Connection Factory properties.

The following table lists the properties available to enable Secure Network Connection (SNC):

Table 14. Secure Network connection properties

Authentication mechanism	Properties
Secure Network Connection (SNC)	sncMyname, sncPartnername, sncQop, sncMode, sncLib

Refer to “Enable Secure Network Connection” on page 228, for more information.

Note: The SSO token / X509 authentication mechanism can also be accomplished by setting the username to "$X509CERT$" (for X509 certificate based authentication) and "$MYSAPSSO2$" (for SSO token based authentication) and setting the password to a base 64 encoded string, which comprises of a SSO token / X509 certificate.

To configure the adapter to create an SAP server connection, see “Passing the connection parameters dynamically.”

Passing the connection parameters dynamically

To pass the connection-related properties dynamically as part of the outbound request you must configure the connection specification class name and set the connection properties on the business graph.

Before you begin

1. WebSphere adapter for SAP import interface, for example, **SAPOutboundInterface**, must be created for the required outbound operations by running the J2C Bean wizard.
2. The input data type for each of the outbound operation must be configured to use the business graph of the business object. For example, the input data type of the operations can be `SapBapiCustomerGetdetailBG` or `SapBapiCustomerCreateBG`.

   The business graph implementation has a child business object, ‘properties' defined as an element in the business graph schema definition. The connection properties must be set in the dataobject ‘properties' of the business graph.

About this task

To pass the connection-related properties dynamically as part of the outbound request, follow this procedure.

Procedure

1. Configure the ConnectionSpec class name in the SAP Import created.
   a. Right-click the SAP adapter import in the assembly diagram and select **Show in** > **Properties view**.
   b. In the Properties tab, select **Binding** > **End-point Configuration**.
   c. In the Connection Spec properties tab, select ConnectionSpec class name as `com.ibm.j2ca.sap.SAPConnectionSpec`
2. Set the **Resource authentication** field in Security Attributes to `Application`.
   a. Select **Security Attributes** from Binding properties.
   b. Set the **Resource authentication** property to Application from Advanced properties. The default value is Container.

When the Resource Authentication property is set to Application, the Java EE component runs a programmatic sign-on to the SAP server. The application component passes security information, such as user name and password, through the ConnectionSpec instance.

3. Set the **Connection properties** in the BusinessGraph within the properties child business object.

   For the adapter to accept the connection parameters dynamically during an outbound request, the application component must set the connection parameters on the business graph data object of the business object.

   The connection properties set on the business graph are prefixed as "CS" to identify them as ConnectionSpec properties. For example, you can set the user name and password to 'CSusername' and 'CSpassword' in the properties element of the BusinessGraph to set the values of connection properties.

   Note: The host name, protocol, or port number values are not accepted through the ConnectionSpec properties. The adapter accepts only authentication-related properties of the user, such as user name, password, SSO token, X509 certificate or SNC properties.

Results

The connection parameters are configured.

What to do next

Create an interface and a Java component, and then deploy the application onto the IBM Business Process Manager.

Configuring the Java Beans 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 business objects and services for BAPI outbound processing

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

Before you begin

Ensure that you have set the connection properties for the J2C Bean wizard.

About this task

Specify the 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 icon.
b. Click the filter button. The Filter Properties window is displayed.

   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 default 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)** to show the objects that match the search criteria of BAPI_CUSTOMER*.
   b. From the Discovered object list, select one or more BAPIs that you want to use.
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 of business objects** check box to generate business object attribute names using the original SAP field names without casing. If you want to generate business object attribute names using the original SAP field name casing, check the **Use the original character case of SAP field names to generate business object attribute names** check box. By default (when the **Use SAP field names to generate attribute names of business objects** check box is not selected), field descriptions are used to generate properties.

      Note: This option is not available when you select the option in the Discovery Configurationwindow.
   b. Optionally, select the **Process Date fields as String** check box. This allows the adapter to retrieve SAP-specific date formats which are otherwise non-standard in Java by using the String type instead of the Date type format.

      Note: To use this feature, ensure that the above check box is selected while running the J2C Bean wizard. As the date fields are converted to string type, ensure that all downstream dependencies (if any) are taken care of.
   c. 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.

If you are adding BAPI_CUSTOMER_GETDETAIL function module, you have the option of adding the following parameters:

Optional Import Parameters

PI_PASS_BUFFER

PI_DIVISION

PI_DISTR_CHAN

Optional Export Parameters:

PE_ADDRESS

RETURN

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

d. 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. If you do not select the **Generate BAPIs within Wrappers** check box, top-level business objects are automatically generated for each BAPI selected. For each top-level business object generated, the adapter internally assigns the **Execute** operation to it. There is no limit on the number of BAPI's that you can configure.

   If you select the **Generate BAPIs within Wrappers** check box, top-level business objects are generated that contain a child business object for each BAPI selected. You can configure a maximum number of four BAPI's.

   Perform one of the following sets of tasks if you have selected **Generate BAPIs within Wrappers** check box:

   - 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 per 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.

   If you enter an invalid namespace value, an error message is displayed as follows: **This property should be a URL value** and the next field becomes disabled. Valid namespace values are URLs that are valid as per the Java that supports it.
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.

   Note: This field cannot be edited if you are modifying existing artifacts.

   Note: If you are creating multiple adapter artifacts within a module, ensure that you specify different business object folders for each adapter within the module. For example, if you are creating artifacts for Oracle, JDBC, SAP, and JDE within a module, you need to create different relative folders for each of these adapters. If you do not specify different relative folders, the existing artifacts are overwritten when you generate new artifacts.

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

   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. To enable the adapter to wait until all time critical updates to SAP database are completed before calling commit on it, select the **Wait until commit call to SAP database is completed and returned** check box. This option is only available if you are using the CWYAP_SAPAdapter.rar.
8. 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 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 project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the project name you want appears in the **Project Selection** list, select its name.
   - Otherwise, click **New** to create a project. In the New Source Project Creation window, select the type of project you want to create.
     a. Java project: For information about creating a Java project, see Creating a Java project.
     b. EJB project: For information about creating a EJB project, see Creating an EJB project.
     c. Web project: For information about creating a web project, see Creating a web project.
2. In the **Package name** field, select or create a package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the package name you want appears in the **Project Selection** list, select its name.
   - Otherwise, create a package:
     a. Click **New**.
     b. In the Java Package window, type a name for the package. For example, myadapteroutboundpkg.
     c. 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: In the **Generate Command Bean** section, select the operation for which you want to generate a command bean. 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.
   - In the managed connection mode, you can specify connection properties in the Java Naming and Directory Interface (JNDI) name of the managed connection factory defined on the application server. The managed connection factory is EIS adapter-specific and contains all of the required

connection information. Managed connection implies that the resource adapter is installed directly on the server and, therefore, the JNDI name of the managed connection factory is visible to all EAR (enterprise application archive) files installed on the application server. To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.

- In the non-managed connection mode, the application communicates directly to the EIS and manages all the connections. Therefore, all of the connection information in this setup is encapsulated within the application. For a J2C bean, this means that all of the connection information is specified in the generated J2C bean. To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.

Note: If you check both choices, the Managed Connection is executed first. If the JNDI name cannot be found, the non-managed connection parameters is used.

6. If you select **Managed Connection (recommended)** check box, you specify the connection properties in the JNDI name of the managed connection factory defined on the application server. You can either select an existing JNDI or create a new one.
   - To select an existing JNDI name,
     a. Click **Browse**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. Select the existing JNDI name in the J2C Connection Factories window. The existing RAR and the JNDI name are used for the connection factory.

     Note: When you try to look up an existing JNDI in the JNDI Lookup Wizard screen, the wizard looks up only the JNDI that is created using the first RAR file. If you have more than one RAR file for the same adapter in the server, you can view only the JNDI generated using the first RAR file. The JNDI generated from the other RAR files are not looked up by the wizard.
   - To create a new one, click **New**.
     a. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     b. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field.
     c. Specify the advanced properties by clicking **Advanced**. The RAR file is installed on IBM WebSphere Application Server and the connection factory is created using the JNDI name specified.
     d. Expand each of the following group sections to review the advanced properties:
        1) Select the **Use load balancing** check box to use load balancing to connect to the SAP system. Load balancing properties which are **Message server host**, **Logon group name** and **SAP system ID** have to be specified in the Additional connection configuration panel under the Advanced tab.
        2) The **Gateway host** is filled in, by default, with the value from the **Host name** field.
        3) A default value of **sapgw00** is filled in for **Gateway service**. If you have more than one gateway server in your SAP configuration, change **sapgw00** to the correct value.

4) Select the **Reset JCO client after client closing the connection handle** check box to invoke the reset method on the JCO client and ensure that the changes on the SAP EIS are reflected in the client during an outbound transaction.

5) If you select **Wait until commit call to SAP database is completed and returned** during configuration, the adapter will wait until all time critical updates to SAP database are completed before invoking the commit call. You can set the value for the `Wait on a BAPI Commit call` to be either a non-space character, which is interpreted as 'True', or set it a space character, which is interpreted as 'False.' If you set the value to be 'True', then the BAPI call will wait indefinitely until the update process is completed before exiting.

6) If you are using Secure Network Connection, select **Enable Secure Network Connection**. Enter information in the associated fields (name, partner, security level, and library path). If you select X509 certificate based authentication, the X509 certificate name has to be entered. Secure Network Connection has to be enabled to use X509 certiicate based authentication.

7) If you are using Secure Network Connection, select the **Secure Network Connection (SNC)** check box. Then enter information in the associated fields (name, partner, security level, and library path). Optionally type the name of an X509 certificate.

8) To set RFC and JCo tracing properties, perform the following steps:

   a) Expand **SAP RFC trace configuration** and select **RFC trace on**.

   b) Select the **Write JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API trace logged into the broker trace file will have the string `[JCoAPI]`. All JCo API traces are logged at the `INFO` message log level.

   c) Select a tracing level from the **RFC trace level** list.

   d) Click **Browse** and select a location to which the RFC trace files will be saved.

9) To change the location of the log files for the wizard or the amount of information included in the logs, provide the following information:

   – If you have multiple instances of the adapter, set **Adapter ID** to a value that is unique for this instance.

   – If you want to disguise potentially sensitive user information in log and trace files, select the **Disguise user data as "XXX" in log and trace files** check box.

   See "Managed connection factory properties" on page 219 for more information about these properties.

e. Click **Finish** to return to the J2C Bean Creation and Deployment Configuration window.

7. If you select the **Non-managed Connection** check box, the system connection information area is expanded to show the connection information. Review the connection information and change the values if required. Specify the advanced properties by clicking **Advanced**. Refer to the advanced properties defined for managed connection.
8. When you are finished setting properties, select the **Click to launch the J2C deployment** checkbox to immediately start the wizard.
9. 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 Service Bean directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer for WebSphere Software provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to theRational Application Developer for WebSphere Software documentation.

Generating the EJB or JSP project

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

About this task

You can create a JSP, an EJB, or a web service to deploy your J2C bean.

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 the Enter key.
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**, **Simple JSP** or **Web Service** and click **Next**.

   Note: In the Deployment Information window, the **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 Enterprise bean creation wizard is displayed. This wizard creates the Java project as an EJB. For information about creating an EJB, see Deploying to an EJB.

   Note: You can deploy an EJB only when the J2C bean is in the EJB project. Otherwise, this option is not available for selection.
9. If you select **Simple JSP**, the Simple JSP Creation wizard is displayed. For details about creating a new web project, see Creating a web project. For information about creating a Simple JSP, see Deploying to a Simple JSP.
10. If you select **Web Service**, the Web Service Creation wizard is displayed. For information about creating a web service, Deploying a J2C application as a web service.
11. 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 Discovery Configuration 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.
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. From the Discovered object list, select one or more BAPIs that you want to use.
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 the original character case of SAP field names to generate business object attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   b. Optionally, select the **Process Date fields as String** check box. This allows the adapter to retrieve SAP-specific date formats which are otherwise non-standard in Java by using the String type instead of the Date type format.

      Note: To use this feature, ensure that the above check box is selected while running the J2C Bean wizard. As the date fields are converted to string type, ensure that all downstream dependencies (if any) are taken care of.
   c. 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.

      If you are adding BAPI_CUSTOMER_GETDETAIL function module, you have the option of adding the following parameters:

      Optional Import Parameters:

      PI_PASS_BUFFER

      PI_DIVISION

      PI_DISTR_CHAN

      Optional Export Parameters:

      PE_ADDRESS

      RETURN

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

   If you enter an invalid namespace value, an error message is displayed as follows: **This property should be a URL value** and the next field becomes disabled. Valid namespace values are URLs that are valid as per the Java that supports it.
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. Adapter uses BAPI_TRANSACTION_COMMIT functional module to call the commit on to SAP. If you want BAPI_TRANSACTION_COMMIT function call to wait until all the time-critical (V1) updates are completed, select the **Use wait parameter before calling BAPI commit** check box.
7. 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 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 project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the project name you want appears in the **Project Selection** list, select its name.
   - Otherwise, click **New** to create a project. In the New Source Project Creation window, select the type of project you want to create.
     a. Java project: For information about creating a Java project, see Creating a Java project.
     b. EJB project: For information about creating a EJB project, see Creating an EJB project.
     c. Web project: For information about creating a web project, see Creating a web project.
2. In the **Package name** field, select or create a package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the package name you want appears in the **Project Selection** list, select its name.
   - Otherwise, create a package:
     a. Click **New**.
     b. In the Java Package window, type a name for the package. For example, myadapteroutboundpkg.
     c. 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: In the **Generate Command Bean** section, select the operation for which you want to generate a command bean. 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.
   - In the managed connection mode, you can specify connection properties in the Java Naming and Directory Interface (JNDI) name of the managed connection factory defined on the application server. The managed connection factory is EIS adapter-specific and contains all of the required connection information. Managed connection implies that the resource adapter is installed directly on the server and, therefore, the JNDI name of the managed connection factory is visible to all EAR (enterprise application archive) files installed on the application server. To obtain the connection

through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.

- In the non-managed connection mode, the application communicates directly to the EIS and manages all the connections. Therefore, all of the connection information in this setup is encapsulated within the application. For a J2C bean, this means that all of the connection information is specified in the generated J2C bean. To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.

Note: If you check both choices, the Managed Connection is executed first. If the JNDI name cannot be found, the non-managed connection parameters is used.

6. If you select **Managed Connection (recommended)** check box, you specify the connection properties in the JNDI name of the managed connection factory defined on the application server. You can either select an existing JNDI or create a new one.
   - To select an existing JNDI name,
     a. Click **Browse**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. Select the existing JNDI name in the J2C Connection Factories window. The existing RAR and the JNDI name are used for the connection factory.

     Note: When you try to look up an existing JNDI in the JNDI Lookup Wizard screen, the wizard looks up only the JNDI that is created using the first RAR file. If you have more than one RAR file for the same adapter in the server, you can view only the JNDI generated using the first RAR file. The JNDI generated from the other RAR files are not looked up by the wizard.
   - To create a new one, click **New**.
     a. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     b. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field.
     c. Specify the advanced properties by clicking **Advanced**. The RAR file is installed on IBM WebSphere Application Server and the connection factory is created using the JNDI name specified.
     d. Expand each of the following group sections to review the advanced properties:
        1) Select the **Use load balancing** check box to use load balancing to connect to the SAP system. Load balancing properties which are **Message server host**, **Logon group name** and **SAP system ID** have to be specified in the Additional connection configuration panel under the Advanced tab.
        2) The **Gateway host** is filled in, by default, with the value from the **Host name** field.
        3) A default value of **sapgw00** is filled in for **Gateway service**. If you have more than one gateway server in your SAP configuration, change **sapgw00** to the correct value.
        4) Select the **Reset JCO client after client closing the connection handle** check box to invoke the reset method on the JCO client and ensure that the changes on the SAP EIS are reflected in the client during an outbound transaction.

5) If you select **Wait until commit call to SAP database is completed and returned** during configuration, the adapter will wait until all time critical updates to SAP database are completed before invoking the commit call. You can set the value for the `Wait on a BAPI Commit call` to be either a non-space character, which is interpreted as 'True', or set it a space character, which is interpreted as 'False.' If you set the value to be 'True', then the BAPI call will wait indefinitely until the update process is completed before exiting.

6) If you are using Secure Network Connection, select **Enable Secure Network Connection**. Enter information in the associated fields (name, partner, security level, and library path). If you select X509 certificate based authentication, the X509 certificate name has to be entered. Secure Network Connection has to be enabled to use X509 certiicate based authentication.

7) If you are using Secure Network Connection, select the **Secure Network Connection (SNC)** check box. Then enter information in the associated fields (name, partner, security level, and library path). Optionally type the name of an X509 certificate.

8) To set RFC and JCo tracing properties, perform the following steps:

   a) Expand **SAP RFC trace configuration** and select **RFC trace on**.

   b) Select the **Write JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API trace logged into the broker trace file will have the string `[JCoAPI]`. All JCo API traces are logged at the `INFO` message log level.

   c) Select a tracing level from the **RFC trace level** list.

   d) Click **Browse** and select a location to which the RFC trace files will be saved.

9) To change the location of the log files for the wizard or the amount of information included in the logs, provide the following information:

   – If you have multiple instances of the adapter, set **Adapter ID** to a value that is unique for this instance.

   – If you want to disguise potentially sensitive user information in log and trace files, select the **Disguise user data as "XXX" in log and trace files** check box.

   See "Managed connection factory properties" on page 219 for more information about these properties.

e. Click **Finish** to return to the J2C Bean Creation and Deployment Configuration window.

7. If you select the **Non-managed Connection** check box, the system connection information area is expanded to show the connection information. Review the connection information and change the values if required. Specify the advanced properties by clicking **Advanced**. Refer to the advanced properties defined for managed connection.
8. When you are finished setting properties, select the **Click to launch the J2C deployment** checkbox to immediately start the wizard.
9. 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 Service Bean directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer for WebSphere Software provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to theRational Application Developer for WebSphere Software documentation.

Generating the EJB or JSP project

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

About this task

You can create a JSP, an EJB, or a web service to deploy your J2C bean.

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 the Enter key.
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**, **Simple JSP** or **Web Service** and click **Next**.

   Note: In the Deployment Information window, the **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 Enterprise bean creation wizard is displayed. This wizard creates the Java project as an EJB. For information about creating an EJB, see Deploying to an EJB.

   Note: You can deploy an EJB only when the J2C bean is in the EJB project. Otherwise, this option is not available for selection.
9. If you select **Simple JSP**, the Simple JSP Creation wizard is displayed. For details about creating a new web project, see Creating a web project. For information about creating a Simple JSP, see Deploying to a Simple JSP.
10. If you select **Web Service**, the Web Service Creation wizard is displayed. For information about creating a web service, Deploying a J2C application as a web service.
11. 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 the original character case of SAP field names to generate business object attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
   b. Optionally, select the **Process Date fields as String** check box. This allows the adapter to retrieve SAP-specific date formats which are otherwise non-standard in Java by using the String type instead of the Date type format.

      Note: To use this feature, ensure that the above check box is selected while running the J2C Bean wizard. As the date fields are converted to string type, ensure that all downstream dependencies (if any) are taken care of.
   c. 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.

      If you are adding BAPI_CUSTOMER_GETDETAIL function module, you have the option of adding the following parameters:

      Optional Import Parameters:

      PI_PASS_BUFFER

      PI_DIVISION

      PI_DISTR_CHAN

      Optional Export Parameters:

      PE_ADDRESS

      RETURN

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

   If you enter an invalid namespace value, an error message is displayed as follows: **This property should be a URL value** and the next field becomes disabled. Valid namespace values are URLs that are valid as per the Java that supports it.
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 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 project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the project name you want appears in the **Project Selection** list, select its name.
   - Otherwise, click **New** to create a project. In the New Source Project Creation window, select the type of project you want to create.
     a. Java project: For information about creating a Java project, see Creating a Java project.
     b. EJB project: For information about creating a EJB project, see Creating an EJB project.
     c. Web project: For information about creating a web project, see Creating a web project.
2. In the **Package name** field, select or create a package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the package name you want appears in the **Project Selection** list, select its name.
   - Otherwise, create a package:
     a. Click **New**.
     b. In the Java Package window, type a name for the package. For example, myadapteroutboundpkg.
     c. 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: In the **Generate Command Bean** section, select the operation for which you want to generate a command bean. 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.
   - In the managed connection mode, you can specify connection properties in the Java Naming and Directory Interface (JNDI) name of the managed connection factory defined on the application server. The managed connection factory is EIS adapter-specific and contains all of the required connection information. Managed connection implies that the resource adapter is installed directly on the server and, therefore, the JNDI name of the managed connection factory is visible to all EAR (enterprise application archive) files installed on the application server. To obtain the connection

through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.

- In the non-managed connection mode, the application communicates directly to the EIS and manages all the connections. Therefore, all of the connection information in this setup is encapsulated within the application. For a J2C bean, this means that all of the connection information is specified in the generated J2C bean. To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.

Note: If you check both choices, the Managed Connection is executed first. If the JNDI name cannot be found, the non-managed connection parameters is used.

6. If you select **Managed Connection (recommended)** check box, you specify the connection properties in the JNDI name of the managed connection factory defined on the application server. You can either select an existing JNDI or create a new one.
   - To select an existing JNDI name,
     a. Click **Browse**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. Select the existing JNDI name in the J2C Connection Factories window. The existing RAR and the JNDI name are used for the connection factory.

     Note: When you try to look up an existing JNDI in the JNDI Lookup Wizard screen, the wizard looks up only the JNDI that is created using the first RAR file. If you have more than one RAR file for the same adapter in the server, you can view only the JNDI generated using the first RAR file. The JNDI generated from the other RAR files are not looked up by the wizard.
   - To create a new one, click **New**.
     a. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     b. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field.
     c. Specify the advanced properties by clicking **Advanced**. The RAR file is installed on IBM WebSphere Application Server and the connection factory is created using the JNDI name specified.
     d. Expand each of the following group sections to review the advanced properties:
        1) Select the **Use load balancing** check box to use load balancing to connect to the SAP system. Load balancing properties which are **Message server host**, **Logon group name** and **SAP system ID** have to be specified in the Additional connection configuration panel under the Advanced tab.
        2) The **Gateway host** is filled in, by default, with the value from the **Host name** field.
        3) A default value of **sapgw00** is filled in for **Gateway service**. If you have more than one gateway server in your SAP configuration, change **sapgw00** to the correct value.
        4) Select the **Reset JCO client after client closing the connection handle** check box to invoke the reset method on the JCO client and ensure that the changes on the SAP EIS are reflected in the client during an outbound transaction.

5) If you select **Wait until commit call to SAP database is completed and returned** during configuration, the adapter will wait until all time critical updates to SAP database are completed before invoking the commit call. You can set the value for the `Wait on a BAPI Commit call` to be either a non-space character, which is interpreted as 'True', or set it a space character, which is interpreted as 'False.' If you set the value to be 'True', then the BAPI call will wait indefinitely until the update process is completed before exiting.

6) If you are using Secure Network Connection, select **Enable Secure Network Connection**. Enter information in the associated fields (name, partner, security level, and library path). If you select X509 certificate based authentication, the X509 certificate name has to be entered. Secure Network Connection has to be enabled to use X509 certiicate based authentication.

7) If you are using Secure Network Connection, select the **Secure Network Connection (SNC)** check box. Then enter information in the associated fields (name, partner, security level, and library path). Optionally type the name of an X509 certificate.

8) To set RFC and JCo tracing properties, perform the following steps:

   a) Expand **SAP RFC trace configuration** and select **RFC trace on**.

   b) Select the **Write JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API trace logged into the broker trace file will have the string `[JCoAPI]`. All JCo API traces are logged at the `INFO` message log level.

   c) Select a tracing level from the **RFC trace level** list.

   d) Click **Browse** and select a location to which the RFC trace files will be saved.

9) To change the location of the log files for the wizard or the amount of information included in the logs, provide the following information:

   – If you have multiple instances of the adapter, set **Adapter ID** to a value that is unique for this instance.

   – If you want to disguise potentially sensitive user information in log and trace files, select the **Disguise user data as "XXX" in log and trace files** check box.

   See "Managed connection factory properties" on page 219 for more information about these properties.

e. Click **Finish** to return to the J2C Bean Creation and Deployment Configuration window.

7. If you select the **Non-managed Connection** check box, the system connection information area is expanded to show the connection information. Review the connection information and change the values if required. Specify the advanced properties by clicking **Advanced**. Refer to the advanced properties defined for managed connection.
8. When you are finished setting properties, select the **Click to launch the J2C deployment** checkbox to immediately start the wizard.
9. 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 Service Bean directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer for WebSphere Software provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to theRational Application Developer for WebSphere Software documentation.

Generating the EJB or JSP project

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

About this task

You can create a JSP, an EJB, or a web service to deploy your J2C bean.

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 the Enter key.
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**, **Simple JSP** or **Web Service** and click **Next**.

   Note: In the Deployment Information window, the **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 Enterprise bean creation wizard is displayed. This wizard creates the Java project as an EJB. For information about creating an EJB, see Deploying to an EJB.

   Note: You can deploy an EJB only when the J2C bean is in the EJB project. Otherwise, this option is not available for selection.
9. If you select **Simple JSP**, the Simple JSP Creation wizard is displayed. For details about creating a new web project, see Creating a web project. For information about creating a Simple JSP, see Deploying to a Simple JSP.
10. If you select **Web Service**, the Web Service Creation wizard is displayed. For information about creating a web service, Deploying a J2C application as a web service.
11. Export the project as an EAR file for deployment.

Configuring the Java bean 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 101.
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. From the Discovered object list, 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**. By default, when the check box is not selected, field descriptions are used to generate properties. If you choose to use SAP field names to generate attribute names, two more check boxes are enabled:
   b. Optionally select the **Use SAP field names to generate attribute names** check box. By default (when the first check box is not selected), field descriptions are used to generate properties.
   c. Select the **Use SAP-original casing for Control Record Business object attribute names** check box to generate attribute names in SAP original casing. If unchecked, attribute names are generated using field names from SAP in camel-casing.
   d. Select the **Use SAP-original casing for Data Record Business object attribute names** check box to generate attribute names in SAP original casing. If unchecked, attribute names are generated using field Name from SAP in camel-casing.

      The following are the different possible combinations:

Table 15.

Scenario	Use SAP field names to generate attribute names (check-box)	Use SAP-original casing Control Record business object attribute names (check-box)	Use SAP-original casing Data Record business object attribute names (check-box)	Control Record	Data Record
1	Checked	Checked	Checked	Attribute names are generated using Field Name from SAP (Original SAP casing)	Attribute names are generated using Field Name from SAP (Original SAP casing)
2	Checked	Checked	Not checked	Attribute names are generated using Field Name from SAP (Original SAP casing)	Attribute names are generated using Field Name from SAP (Camel-casing)
3	Checked	Not checked	Checked	Attribute names are generated using Field Name from SAP (Camel-casing)	Attribute names are generated using Field Name from SAP (Original SAP casing)
4	Checked	Not checked	Not checked	Attribute names are generated using Field Name from SAP (Camel-casing)	Attribute names are generated using Field Name from SAP (Camel-casing)
5	Not checked	Not checked	Not checked	Attribute names are generated using Field Description from SAP (Camel-casing)	Attribute names are generated using Field Description from SAP (Camel-casing)

e. 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.

f. 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. If the selected IDoc has unreleased segments, the **IDoc release**

version property becomes required. It is recommended that you select the default value **Unreleased** if the IDoc you are working with has unreleased segments. If **Unreleased** is selected, the adapter generates the business objects for segments using the unreleased segment definition.

g. 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. The version of the SAP system is set as default. If the selected IDoc has unreleased segments, the **IDoc release version** property becomes required. It is recommended that you select **Unreleased** if the IDoc you are working with has unreleased segments. If **Unreleased** is selected, the adapter generates the business objects for segments using the unreleased segment definition.

h. 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 on page 104.
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**. By default, when the check box is not selected, field descriptions are used to generate properties. If you choose to use SAP field names to generate attribute names, two more check boxes are enabled:
   b. Select the **Use SAP-original casing for Control Record Business object attribute names** check box to generate attribute names in SAP original casing. If unchecked, attribute names are generated using field names from SAP in camel-casing.
   c. Select the **Use SAP-original casing for Data Record Business object attribute names** check box to generate attribute names in SAP original casing. If unchecked, attribute names are generated using field Name from SAP in camel-casing.
      The following are the different possible combinations:

Table 16.

Scenario	Use SAP field names to generate attribute names (check-box)	Use SAP-original casing Control Record business object attribute names (check-box)	Use SAP-original casing Data Record business object attribute names (check-box)	Control Record	Data Record
1	Checked	Checked	Checked	Attribute names are generated using Field Name from SAP (Original SAP casing)	Attribute names are generated using Field Name from SAP (Original SAP casing)
2	Checked	Checked	Not checked	Attribute names are generated using Field Name from SAP (Original SAP casing)	Attribute names are generated using Field Name from SAP (Camel-casing)
3	Checked	Not checked	Checked	Attribute names are generated using Field Name from SAP (Camel-casing)	Attribute names are generated using Field Name from SAP (Original SAP casing)

Table 16. (continued)

Scenario	Use SAP field names to generate attribute names (check-box)	Use SAP-original casing Control Record business object attribute names (check-box)	Use SAP-original casing Data Record business object attribute names (check-box)	Control Record	Data Record
4	Checked	Not checked	Not checked	Attribute names are generated using Field Name from SAP (Camel-casing)	Attribute names are generated using Field Name from SAP (Camel-casing)
5	Not checked	Not checked	Not checked	Attribute names are generated using Field Description from SAP (Camel-casing)	Attribute names are generated using Field Description from SAP (Camel-casing)

d. 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.

e. 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.

f. 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.

   If you enter an invalid namespace value, an error message is displayed as follows: **This property should be a URL value** and the next field becomes disabled. Valid namespace values are URLs that are valid as per the Java that supports it.
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.

   Note: This field cannot be edited if you are modifying existing artifacts.

   Note: If you are creating multiple adapter artifacts within a module, ensure that you specify different business object folders for each adapter within the module. For example, if you are creating artifacts for Oracle, JDBC, SAP, and JDE within a module, you need to create different relative folders for each of these adapters. If you do not specify different relative folders, the existing artifacts are overwritten when you generate new artifacts.
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 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 project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the project name you want appears in the **Project Selection** list, select its name.
   - Otherwise, click **New** to create a project. In the New Source Project Creation window, select the type of project you want to create.

a. Java project: For information about creating a Java project, see Creating a Java project.
b. EJB project: For information about creating a EJB project, see Creating an EJB project.
c. Web project: For information about creating a web project, see Creating a web project.

2. In the **Package name** field, select or create a package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the package name you want appears in the **Project Selection** list, select its name.
   - Otherwise, create a package:
     a. Click **New**.
     b. In the Java Package window, type a name for the package. For example, myadapteroutboundpkg.
     c. 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: In the **Generate Command Bean** section, select the operation for which you want to generate a command bean. 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.
   - In the managed connection mode, you can specify connection properties in the Java Naming and Directory Interface (JNDI) name of the managed connection factory defined on the application server. The managed connection factory is EIS adapter-specific and contains all of the required connection information. Managed connection implies that the resource adapter is installed directly on the server and, therefore, the JNDI name of the managed connection factory is visible to all EAR (enterprise application archive) files installed on the application server. To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.
   - In the non-managed connection mode, the application communicates directly to the EIS and manages all the connections. Therefore, all of the connection information in this setup is encapsulated within the application. For a J2C bean, this means that all of the connection information is specified in the generated J2C bean. To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.

   Note: If you check both choices, the Managed Connection is executed first. If the JNDI name cannot be found, the non-managed connection parameters is used.
6. If you select **Managed Connection (recommended)** check box, you specify the connection properties in the JNDI name of the managed connection factory defined on the application server. You can either select an existing JNDI or create a new one.
   - To select an existing JNDI name,
     a. Click **Browse**.

b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
c. Select the existing JNDI name in the J2C Connection Factories window. The existing RAR and the JNDI name are used for the connection factory.

Note: When you try to look up an existing JNDI in the JNDI Lookup Wizard screen, the wizard looks up only the JNDI that is created using the first RAR file. If you have more than one RAR file for the same adapter in the server, you can view only the JNDI generated using the first RAR file. The JNDI generated from the other RAR files are not looked up by the wizard.

- To create a new one, click **New**.
  a. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
  b. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field.
  c. Specify the advanced properties by clicking **Advanced**. The RAR file is installed on IBM WebSphere Application Server and the connection factory is created using the JNDI name specified.
  d. Expand each of the following group sections to review the advanced properties:
     1) Select the **Use load balancing** check box to use load balancing to connect to the SAP system. Load balancing properties which are **Message server host**, **Logon group name** and **SAP system ID** have to be specified in the Additional connection configuration panel under the Advanced tab.
     2) The **Gateway host** is filled in, by default, with the value from the **Host name** field.
     3) A default value of **sapgw00** is filled in for **Gateway service**. If you have more than one gateway server in your SAP configuration, change **sapgw00** to the correct value.
     4) Select the **Reset JCO client after client closing the connection handle** check box to invoke the reset method on the JCO client and ensure that the changes on the SAP EIS are reflected in the client during an outbound transaction.
     5) If you select **Wait until commit call to SAP database is completed and returned** during configuration, the adapter will wait until all time critical updates to SAP database are completed before invoking the commit call. You can set the value for the `Wait on a BAPI Commit call` to be either a non-space character, which is interpreted as 'True', or set it a space character, which is interpreted as 'False.' If you set the value to be 'True', then the BAPI call will wait indefinitely until the update process is completed before exiting.
     6) If you are using Secure Network Connection, select **Enable Secure Network Connection**. Enter information in the associated fields (name, partner, security level, and library path). If you select X509 certificate based authentication, the X509 certificate name has to be entered. Secure Network Connection has to be enabled to use X509 certiicate based authentication.
     7) If you are using Secure Network Connection, select the **Secure Network Connection (SNC)** check box. Then enter information in the associated fields (name, partner, security level, and library path). Optionally type the name of an X509 certificate.

8) To set RFC and JCo tracing properties, perform the following steps:
   a) Expand **SAP RFC trace configuration** and select **RFC trace on**.
   b) Select the **Write JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API trace logged into the broker trace file will have the string `[JCoAPI]`. All JCo API traces are logged at the `INFO` message log level.
   c) Select a tracing level from the **RFC trace level** list.
   d) Click **Browse** and select a location to which the RFC trace files will be saved.
9) To change the location of the log files for the wizard or the amount of information included in the logs, provide the following information:
   – If you have multiple instances of the adapter, set **Adapter ID** to a value that is unique for this instance.
   – If you want to disguise potentially sensitive user information in log and trace files, select the **Disguise user data as "XXX" in log and trace files** check box.

   See "Managed connection factory properties" on page 219 for more information about these properties.

e. Click **Finish** to return to the J2C Bean Creation and Deployment Configuration window.

7. If you select the **Non-managed Connection** check box, the system connection information area is expanded to show the connection information. Review the connection information and change the values if required. Specify the advanced properties by clicking **Advanced**. Refer to the advanced properties defined for managed connection.
8. When you are finished setting properties, select the **Click to launch the J2C deployment** checkbox to immediately start the wizard.
9. 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 Service Bean directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer for WebSphere Software provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to theRational Application Developer for WebSphere Software documentation.

Generating the EJB or JSP project

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

About this task

You can create a JSP, an EJB, or a web service to deploy your J2C bean.

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 the Enter key.
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**, **Simple JSP** or **Web Service** and click **Next**.

   Note: In the Deployment Information window, the **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 Enterprise bean creation wizard is displayed. This wizard creates the Java project as an EJB. For information about creating an EJB, see Deploying to an EJB.

   Note: You can deploy an EJB only when the J2C bean is in the EJB project. Otherwise, this option is not available for selection.
9. If you select **Simple JSP**, the Simple JSP Creation wizard is displayed. For details about creating a new web project, see Creating a web project. For information about creating a Simple JSP, see Deploying to a Simple JSP.
10. If you select **Web Service**, the Web Service Creation wizard is displayed. For information about creating a web service, Deploying a J2C application as a web service.
11. 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.
   - If your Data Record size to be less than the standard length (1063 characters) specified by SAP adapter, use a delimiter value in the **Delimiter to be used for splitting multiple IDocs** field. Any valid string without escape characters or either of the two escape characters, **\n** or **\r\n** can be used as a delimiter value. For more information on delimiters refer to "Outbound processing for the ALE pass-through IDoc interface" on page 42.

     If your hexbinary content has non-unicode characters, set the partner character set to the specific character set in the following steps: 5
4. Click **OK**.
5. 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 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 project into which the J2C Bean is generated.

- To select an existing project, click **Browse**. If the project name you want appears in the **Project Selection** list, select its name.
- Otherwise, click **New** to create a project. In the New Source Project Creation window, select the type of project you want to create.
  a. Java project: For information about creating a Java project, see Creating a Java project.
  b. EJB project: For information about creating a EJB project, see Creating an EJB project.
  c. Web project: For information about creating a web project, see Creating a web project.

2. In the **Package name** field, select or create a package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the package name you want appears in the **Project Selection** list, select its name.
   - Otherwise, create a package:
     a. Click **New**.
     b. In the Java Package window, type a name for the package. For example, myadapteroutboundpkg.
     c. 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: In the **Generate Command Bean** section, select the operation for which you want to generate a command bean. 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.
   - In the managed connection mode, you can specify connection properties in the Java Naming and Directory Interface (JNDI) name of the managed connection factory defined on the application server. The managed connection factory is EIS adapter-specific and contains all of the required connection information. Managed connection implies that the resource adapter is installed directly on the server and, therefore, the JNDI name of the managed connection factory is visible to all EAR (enterprise application archive) files installed on the application server. To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.
   - In the non-managed connection mode, the application communicates directly to the EIS and manages all the connections. Therefore, all of the connection information in this setup is encapsulated within the application. For a J2C bean, this means that all of the connection information is specified in the generated J2C bean. To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.

   Note: If you check both choices, the Managed Connection is executed first. If the JNDI name cannot be found, the non-managed connection parameters is used.

6. If you select **Managed Connection (recommended)** check box, you specify the connection properties in the JNDI name of the managed connection factory defined on the application server. You can either select an existing JNDI or create a new one.
   - To select an existing JNDI name,
     a. Click **Browse**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. Select the existing JNDI name in the J2C Connection Factories window. The existing RAR and the JNDI name are used for the connection factory.

     Note: When you try to look up an existing JNDI in the JNDI Lookup Wizard screen, the wizard looks up only the JNDI that is created using the first RAR file. If you have more than one RAR file for the same adapter in the server, you can view only the JNDI generated using the first RAR file. The JNDI generated from the other RAR files are not looked up by the wizard.
   - To create a new one, click **New**.
     a. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     b. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field.
     c. Specify the advanced properties by clicking **Advanced**. The RAR file is installed on IBM WebSphere Application Server and the connection factory is created using the JNDI name specified.
     d. Expand each of the following group sections to review the advanced properties:
        1) Select the **Use load balancing** check box to use load balancing to connect to the SAP system. Load balancing properties which are **Message server host**, **Logon group name** and **SAP system ID** have to be specified in the Additional connection configuration panel under the Advanced tab.
        2) The **Gateway host** is filled in, by default, with the value from the **Host name** field.
        3) A default value of **sapgw00** is filled in for **Gateway service**. If you have more than one gateway server in your SAP configuration, change **sapgw00** to the correct value.
        4) Select the **Reset JCO client after client closing the connection handle** check box to invoke the reset method on the JCO client and ensure that the changes on the SAP EIS are reflected in the client during an outbound transaction.
        5) If you select **Wait until commit call to SAP database is completed and returned** during configuration, the adapter will wait until all time critical updates to SAP database are completed before invoking the commit call. You can set the value for the `Wait on a BAPI Commit call` to be either a non-space character, which is interpreted as 'True', or set it a space character, which is interpreted as 'False.' If you set the value to be 'True', then the BAPI call will wait indefinitely until the update process is completed before exiting.
        6) If you are using Secure Network Connection, select **Enable Secure Network Connection**. Enter information in the associated fields (name, partner, security level, and library path). If you select X509

certificate based authentication, the X509 certificate name has to be entered. Secure Network Connection has to be enabled to use X509 certiicate based authentication.

7) If you are using Secure Network Connection, select the **Secure Network Connection (SNC)** check box. Then enter information in the associated fields (name, partner, security level, and library path). Optionally type the name of an X509 certificate.
8) To set RFC and JCo tracing properties, perform the following steps:
   a) Expand **SAP RFC trace configuration** and select **RFC trace on**.
   b) Select the **Write JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API trace logged into the broker trace file will have the string `[JCoAPI]`. All JCo API traces are logged at the `INFO` message log level.
   c) Select a tracing level from the **RFC trace level** list.
   d) Click **Browse** and select a location to which the RFC trace files will be saved.
9) To change the location of the log files for the wizard or the amount of information included in the logs, provide the following information:
   – If you have multiple instances of the adapter, set **Adapter ID** to a value that is unique for this instance.
   – If you want to disguise potentially sensitive user information in log and trace files, select the **Disguise user data as "XXX" in log and trace files** check box.

   See "Managed connection factory properties" on page 219 for more information about these properties.

e. Click **Finish** to return to the J2C Bean Creation and Deployment Configuration window.

7. If you select the **Non-managed Connection** check box, the system connection information area is expanded to show the connection information. Review the connection information and change the values if required. Specify the advanced properties by clicking **Advanced**. Refer to the advanced properties defined for managed connection.
8. When you are finished setting properties, select the **Click to launch the J2C deployment** checkbox to immediately start the wizard.
9. 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 Service Bean directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer for WebSphere Software provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to theRational Application Developer for WebSphere Software documentation.

Generating the EJB or JSP project

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

About this task

You can create a JSP, an EJB, or a web service to deploy your J2C bean.

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 the Enter key.
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**, **Simple JSP** or **Web Service** and click **Next**.

   Note: In the Deployment Information window, the **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 Enterprise bean creation wizard is displayed. This wizard creates the Java project as an EJB. For information about creating an EJB, see Deploying to an EJB.

   Note: You can deploy an EJB only when the J2C bean is in the EJB project. Otherwise, this option is not available for selection.
9. If you select **Simple JSP**, the Simple JSP Creation wizard is displayed. For details about creating a new web project, see Creating a web project. For information about creating a Simple JSP, see Deploying to a Simple JSP.
10. If you select **Web Service**, the Web Service Creation wizard is displayed. For information about creating a web service, Deploying a J2C application as a web service.
11. 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 Discovery Configuration 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 Parameters *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 Parameters *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.

   If you enter an invalid namespace value, an error message is displayed as follows: **This property should be a URL value** and the next field becomes disabled. Valid namespace values are URLs that are valid as per the Java that supports it.
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.

   Note: This field cannot be edited if you are modifying existing artifacts.

   Note: If you are creating multiple adapter artifacts within a module, ensure that you specify different business object folders for each adapter within the module. For example, if you are creating artifacts for Oracle, JDBC, SAP, and JDE within a module, you need to create different relative folders for each of these adapters. If you do not specify different relative folders, the existing artifacts are overwritten when you generate new artifacts.
3. In the **Custom retrieve function name** field, type the name of the function you created on the EIS to avoid the improper data truncation with delimiter set.
4. Click **Next** to continue to the Service Generation and Deployment Configuration window.

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 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 project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the project name you want appears in the **Project Selection** list, select its name.
   - Otherwise, click **New** to create a project. In the New Source Project Creation window, select the type of project you want to create.
     a. Java project: For information about creating a Java project, see Creating a Java project.
     b. EJB project: For information about creating a EJB project, see Creating an EJB project.
     c. Web project: For information about creating a web project, see Creating a web project.
2. In the **Package name** field, select or create a package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the package name you want appears in the **Project Selection** list, select its name.
   - Otherwise, create a package:
     a. Click **New**.
     b. In the Java Package window, type a name for the package. For example, myadapteroutboundpkg.
     c. 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: In the **Generate Command Bean** section, select the operation for which you want to generate a command bean. 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.
   - In the managed connection mode, you can specify connection properties in the Java Naming and Directory Interface (JNDI) name of the managed connection factory defined on the application server. The managed connection factory is EIS adapter-specific and contains all of the required connection information. Managed connection implies that the resource adapter is installed directly on the server and, therefore, the JNDI name of the managed connection factory is visible to all EAR (enterprise application archive) files installed on the application server. To obtain the connection

through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.

- In the non-managed connection mode, the application communicates directly to the EIS and manages all the connections. Therefore, all of the connection information in this setup is encapsulated within the application. For a J2C bean, this means that all of the connection information is specified in the generated J2C bean. To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.

Note: If you check both choices, the Managed Connection is executed first. If the JNDI name cannot be found, the non-managed connection parameters is used.

6. If you select **Managed Connection (recommended)** check box, you specify the connection properties in the JNDI name of the managed connection factory defined on the application server. You can either select an existing JNDI or create a new one.
   - To select an existing JNDI name,
     a. Click **Browse**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. Select the existing JNDI name in the J2C Connection Factories window. The existing RAR and the JNDI name are used for the connection factory.

     Note: When you try to look up an existing JNDI in the JNDI Lookup Wizard screen, the wizard looks up only the JNDI that is created using the first RAR file. If you have more than one RAR file for the same adapter in the server, you can view only the JNDI generated using the first RAR file. The JNDI generated from the other RAR files are not looked up by the wizard.
   - To create a new one, click **New**.
     a. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     b. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field.
     c. Specify the advanced properties by clicking **Advanced**. The RAR file is installed on IBM WebSphere Application Server and the connection factory is created using the JNDI name specified.
     d. Expand each of the following group sections to review the advanced properties:
        1) Select the **Use load balancing** check box to use load balancing to connect to the SAP system. Load balancing properties which are **Message server host**, **Logon group name** and **SAP system ID** have to be specified in the Additional connection configuration panel under the Advanced tab.
        2) The **Gateway host** is filled in, by default, with the value from the **Host name** field.
        3) A default value of **sapgw00** is filled in for **Gateway service**. If you have more than one gateway server in your SAP configuration, change **sapgw00** to the correct value.
        4) Select the **Reset JCO client after client closing the connection handle** check box to invoke the reset method on the JCO client and ensure that the changes on the SAP EIS are reflected in the client during an outbound transaction.

5) If you select **Wait until commit call to SAP database is completed and returned** during configuration, the adapter will wait until all time critical updates to SAP database are completed before invoking the commit call. You can set the value for the `Wait on a BAPI Commit call` to be either a non-space character, which is interpreted as 'True', or set it a space character, which is interpreted as 'False.' If you set the value to be 'True', then the BAPI call will wait indefinitely until the update process is completed before exiting.

6) If you are using Secure Network Connection, select **Enable Secure Network Connection**. Enter information in the associated fields (name, partner, security level, and library path). If you select X509 certificate based authentication, the X509 certificate name has to be entered. Secure Network Connection has to be enabled to use X509 certiicate based authentication.

7) If you are using Secure Network Connection, select the **Secure Network Connection (SNC)** check box. Then enter information in the associated fields (name, partner, security level, and library path). Optionally type the name of an X509 certificate.

8) To set RFC and JCo tracing properties, perform the following steps:

   a) Expand **SAP RFC trace configuration** and select **RFC trace on**.

   b) Select the **Write JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API trace logged into the broker trace file will have the string `[JCoAPI]`. All JCo API traces are logged at the `INFO` message log level.

   c) Select a tracing level from the **RFC trace level** list.

   d) Click **Browse** and select a location to which the RFC trace files will be saved.

9) To change the location of the log files for the wizard or the amount of information included in the logs, provide the following information:

   – If you have multiple instances of the adapter, set **Adapter ID** to a value that is unique for this instance.

   – If you want to disguise potentially sensitive user information in log and trace files, select the **Disguise user data as "XXX" in log and trace files** check box.

   See "Managed connection factory properties" on page 219 for more information about these properties.

e. Click **Finish** to return to the J2C Bean Creation and Deployment Configuration window.

7. If you select the **Non-managed Connection** check box, the system connection information area is expanded to show the connection information. Review the connection information and change the values if required. Specify the advanced properties by clicking **Advanced**. Refer to the advanced properties defined for managed connection.
8. When you are finished setting properties, select the **Click to launch the J2C deployment** checkbox to immediately start the wizard.
9. 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 Service Bean directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer for WebSphere Software provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to theRational Application Developer for WebSphere Software documentation.

Generating the EJB or JSP project

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

About this task

You can create a JSP, an EJB, or a web service to deploy your J2C bean.

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 the Enter key.
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**, **Simple JSP** or **Web Service** and click **Next**.

   Note: In the Deployment Information window, the **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 Enterprise bean creation wizard is displayed. This wizard creates the Java project as an EJB. For information about creating an EJB, see Deploying to an EJB.

   Note: You can deploy an EJB only when the J2C bean is in the EJB project. Otherwise, this option is not available for selection.
9. If you select **Simple JSP**, the Simple JSP Creation wizard is displayed. For details about creating a new web project, see Creating a web project. For information about creating a Simple JSP, see Deploying to a Simple JSP.
10. If you select **Web Service**, the Web Service Creation wizard is displayed. For information about creating a web service, Deploying a J2C application as a web service.
11. 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.
2. From the Filter Propertieswindow, 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. From the Discovered object list, 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.

If you enter an invalid namespace value, an error message is displayed as follows: **This property should be a URL value** and the next field becomes disabled. Valid namespace values are URLs that are valid as per the Java that supports it.

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.

   Note: This field cannot be edited if you are modifying existing artifacts.

   Note: If you are creating multiple adapter artifacts within a module, ensure that you specify different business object folders for each adapter within the module. For example, if you are creating artifacts for Oracle, JDBC, SAP, and JDE within a module, you need to create different relative folders for each of these adapters. If you do not specify different relative folders, the existing artifacts are overwritten when you generate new artifacts.
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 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 project into which the J2C Bean is generated.
   - To select an existing project, click **Browse**. If the project name you want appears in the **Project Selection** list, select its name.
   - Otherwise, click **New** to create a project. In the New Source Project Creation window, select the type of project you want to create.
     a. Java project: For information about creating a Java project, see Creating a Java project.
     b. EJB project: For information about creating a EJB project, see Creating an EJB project.

c. Web project: For information about creating a web project, see Creating a web project.

2. In the **Package name** field, select or create a package into which the J2C Bean is generated.
   - To select an existing package, click **Browse**. If the package name you want appears in the **Project Selection** list, select its name.
   - Otherwise, create a package:
     a. Click **New**.
     b. In the Java Package window, type a name for the package. For example, myadapteroutboundpkg.
     c. 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: In the **Generate Command Bean** section, select the operation for which you want to generate a command bean. 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.
   - In the managed connection mode, you can specify connection properties in the Java Naming and Directory Interface (JNDI) name of the managed connection factory defined on the application server. The managed connection factory is EIS adapter-specific and contains all of the required connection information. Managed connection implies that the resource adapter is installed directly on the server and, therefore, the JNDI name of the managed connection factory is visible to all EAR (enterprise application archive) files installed on the application server. To obtain the connection through JNDI, select the **Managed Connection (recommended)** check box. This type of connection is managed by the application server.
   - In the non-managed connection mode, the application communicates directly to the EIS and manages all the connections. Therefore, all of the connection information in this setup is encapsulated within the application. For a J2C bean, this means that all of the connection information is specified in the generated J2C bean. To obtain the connection directly from the resource adapter, select the **Non-managed Connection** check box.

   Note: If you check both choices, the Managed Connection is executed first. If the JNDI name cannot be found, the non-managed connection parameters is used.
6. If you select **Managed Connection (recommended)** check box, you specify the connection properties in the JNDI name of the managed connection factory defined on the application server. You can either select an existing JNDI or create a new one.
   - To select an existing JNDI name,
     a. Click **Browse**.
     b. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
     c. Select the existing JNDI name in the J2C Connection Factories window. The existing RAR and the JNDI name are used for the connection factory.

Note: When you try to look up an existing JNDI in the JNDI Lookup Wizard screen, the wizard looks up only the JNDI that is created using the first RAR file. If you have more than one RAR file for the same adapter in the server, you can view only the JNDI generated using the first RAR file. The JNDI generated from the other RAR files are not looked up by the wizard.

- To create a new one, click **New**.
  a. In the Server selection window, specify the server on which to deploy the adapter and click **Next**.
  b. In the New J2C Connection Factory window, specify the name in the **JNDI Name** field.
  c. Specify the advanced properties by clicking **Advanced**. The RAR file is installed on IBM WebSphere Application Server and the connection factory is created using the JNDI name specified.
  d. Expand each of the following group sections to review the advanced properties:
     1) Select the **Use load balancing** check box to use load balancing to connect to the SAP system. Load balancing properties which are **Message server host**, **Logon group name** and **SAP system ID** have to be specified in the Additional connection configuration panel under the Advanced tab.
     2) The **Gateway host** is filled in, by default, with the value from the **Host name** field.
     3) A default value of **sapgw00** is filled in for **Gateway service**. If you have more than one gateway server in your SAP configuration, change **sapgw00** to the correct value.
     4) Select the **Reset JCO client after client closing the connection handle** check box to invoke the reset method on the JCO client and ensure that the changes on the SAP EIS are reflected in the client during an outbound transaction.
     5) If you select **Wait until commit call to SAP database is completed and returned** during configuration, the adapter will wait until all time critical updates to SAP database are completed before invoking the commit call. You can set the value for the `Wait on a BAPI Commit call` to be either a non-space character, which is interpreted as 'True', or set it a space character, which is interpreted as 'False.' If you set the value to be 'True', then the BAPI call will wait indefinitely until the update process is completed before exiting.
     6) If you are using Secure Network Connection, select **Enable Secure Network Connection**. Enter information in the associated fields (name, partner, security level, and library path). If you select X509 certificate based authentication, the X509 certificate name has to be entered. Secure Network Connection has to be enabled to use X509 certiicate based authentication.
     7) If you are using Secure Network Connection, select the **Secure Network Connection (SNC)** check box. Then enter information in the associated fields (name, partner, security level, and library path). Optionally type the name of an X509 certificate.
     8) To set RFC and JCo tracing properties, perform the following steps:
        a) Expand **SAP RFC trace configuration** and select **RFC trace on**.
        b) Select the **Write JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API

trace logged into the broker trace file will have the string [JCoAPI]. All JCo API traces are logged at the INFO message log level.

c) Select a tracing level from the **RFC trace level** list.

d) Click **Browse** and select a location to which the RFC trace files will be saved.

9) To change the location of the log files for the wizard or the amount of information included in the logs, provide the following information:

- If you have multiple instances of the adapter, set **Adapter ID** to a value that is unique for this instance.
- If you want to disguise potentially sensitive user information in log and trace files, select the **Disguise user data as "XXX" in log and trace files** check box.

See "Managed connection factory properties" on page 219 for more information about these properties.

e. Click **Finish** to return to the J2C Bean Creation and Deployment Configuration window.

7. If you select the **Non-managed Connection** check box, the system connection information area is expanded to show the connection information. Review the connection information and change the values if required. Specify the advanced properties by clicking **Advanced**. Refer to the advanced properties defined for managed connection.
8. When you are finished setting properties, select the **Click to launch the J2C deployment** checkbox to immediately start the wizard.
9. 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 Service Bean directly, in the non-managed mode or generate JSP or EJB that uses the J2C Bean. Rational Application Developer for WebSphere Software provides tools to automate this generation. You can access these tools from the **New>Others** menu, for their description refer to theRational Application Developer for WebSphere Software documentation.

Generating the EJB or JSP project

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

About this task

You can create a JSP, an EJB, or a web service to deploy your J2C bean.

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 the Enter key.
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**, **Simple JSP** or **Web Service** and click **Next**.

   Note: In the Deployment Information window, the **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 Enterprise bean creation wizard is displayed. This wizard creates the Java project as an EJB. For information about creating an EJB, see Deploying to an EJB.

   Note: You can deploy an EJB only when the J2C bean is in the EJB project. Otherwise, this option is not available for selection.
9. If you select **Simple JSP**, the Simple JSP Creation wizard is displayed. For details about creating a new web project, see Creating a web project. For information about creating a Simple JSP, see Deploying to a Simple JSP.
10. If you select **Web Service**, the Web Service Creation wizard is displayed. For information about creating a web service, Deploying a J2C application as a web service.
11. 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.

Before you begin

The following table outlines the properties that are applicable for an interface:

Table 17.

Interface	SAP RFC Trace Configuration	Bidi properties	Business Objects naming convention	
			SAP PI naming convention	Enforce naming convention for business objects
BAPI	Yes	Yes	Yes	Yes
ALE	Yes	Yes	Yes	No
ALE Pass-through Idoc	Yes	Yes	No	No
Advanced Event Process (AEP)	Yes	Yes	No	No

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.
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. From the Discovered object list, select one or more BAPIs that you want to use.
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 the original character case of SAP field names to generate business object attribute names** check box. By default (when the check box is not selected), the field descriptions are used to generate properties.
b. Optionally, select the **Process Date fields as String** check box. This allows the adapter to retrieve SAP-specific date formats which are otherwise non-standard in Java by using the String type instead of the Date type format.

   Note: To use this feature, ensure that the above check box is selected while running the J2C Bean wizard. As the date fields are converted to string type, ensure that all downstream dependencies (if any) are taken care of.
c. 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.

   If you are adding BAPI_CUSTOMER_GETDETAIL function module, you have the option of adding the following parameters:

   Optional Import Parameters

   PI_PASS_BUFFER

   PI_DIVISION

   PI_DISTR_CHAN

   Optional Export Parameters

   PE_ADDRESS

   RETURN

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

   If you enter an invalid namespace value, an error message is displayed as follows: **This property should be a URL value** and the next field becomes disabled. Valid namespace values are URLs that are valid as per the Java that supports it.
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.

   Note: The above two fields cannot be edited if you are modifying existing artifacts.

   Note: If you are creating multiple adapter artifacts within a module, ensure that you specify different business object folders for each adapter within the module. For example, if you are creating artifacts for Oracle, JDBC, SAP, and JDE within a module, you need to create different relative folders for each of

these adapters. If you do not specify different relative folders, the existing artifacts are overwritten when you generate new artifacts.

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**.

   Note: This field cannot be edited if you are modifying existing artifacts.

5. Click **Next**.

Results

You selected an operation for each BAPI. The J2C Bean Creation 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 the 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 EJB 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 required 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 required 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, select the JNDI name for an existing activation specification in WebSphere Application Server or create an activation specification. For more information about creating an activation specification in WebSphere Application Server, see Setting activation specification properties for stand-alone adapters.
   - To select an existing activation specification, click **Browse**.

     Note: When you try to look up an existing JNDI in the JNDI Lookup Wizard screen, the wizard looks up only the JNDI that is created using the first RAR file. If you have more than one RAR file for the same adapter in the server, you can view only the JNDI generated using the first RAR file. The JNDI generated from the other RAR files are not looked up by the wizard.
   - To create a new activation specification, click **New**.
     a. In the Server selection window, select the server on which the resource adapter will be deployed and click **Next**.
     b. In the New J2C Activation Specification window, type a JNDI name for the activation specification.
     c. To set additional properties, click **Advanced**.
        1) Select the **Use load balancing** check box to use load balancing to connect to the SAP system. Load balancing properties which are **Message server host**, **Logon group name** and **SAP system ID** have to be specified in the Additional connection configuration panel under the Advanced tab.
        2) The **Gateway host** is filled in, by default, with the value from the **Host name** field.
        3) A default value of **sapgw00** is filled in for **Gateway service**. If you have more than one gateway server in your SAP configuration, change **sapgw00** to the correct value.

Note: See "Activation specification properties for BAPI inbound processing" on page 250 for more information.

4) **Event polling configuration**
   a) In **Interval between polling periods**, type the number of milliseconds that the adapter waits between polling periods.
   b) In **Maximum events in polling period**, type the number of events to deliver in each polling period.
   c) In **Time between retries in case of system connection failure (milliseconds)**, specify the time interval between attempts to restart the event listeners.
   d) In **Maximum number of retries in case of system connection failure**, specify the number of times the adapter should try attempt to restart the event listeners.
   e) Select the **Stop the adapter when an error is encountered while polling** check box to specify whether the adapter should stop polling for events when it encounters an error during polling.
   f) Select the **Retry EIS connection on startup** check box to specify if you want the adapter to retry a failed connection when starting.
5) **Event delivery configuration**
   a) Select **Type of delivery** to specify the order in which events are delivered by the adapter to the export.
   b) Select the **Ensure assured-once event delivery** check box to get a once-only delivery of the inbound events. This may reduce performance.
   c) Select the **Do not process events that have a timestamp in the future** to specify whether the adapter should filter out future events by comparing the timestamp on each event with the system time.
   d) Select the **Event types to process** to specify a list of event types that indicates to the adapter which events it should deliver.
   e) Select the **Retry limit for failed events** to specify the number of times that the adapter attempts to re-deliver an event before marking the event as failed.
6) If you are using Secure Network Connection, select the **Secure Network Connection (SNC)** check box. Then enter information in the associated fields (name, partner, security level, and library path). Optionally type the name of an X509 certificate.
7) Optionally to set RFC and JCo tracing properties, perform the following steps:
   a) Expand **SAP RFC trace configuration** and select **RFC trace on**.
   b) Select the **Write JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API trace logged into the broker trace file will have the string `[JCoAPI]`. All JCo API traces are logged at the `INFO` message log level.
   c) Select a tracing level from the **RFC trace level** list.
   d) Click **Browse** and select a location to which the RFC trace files will be saved.
8) **AEP Data Format configuration**

a) Select **IDoc empty tags** to include empty tags to the unpopulated fields in the IDoc segment, which are sent to a configured endpoint, based on the option selected.

See Activation specification properties for more information about these properties.

d. Click **Finish** to return to the J2C Bean Creation and Deployment Configuration window.

5. Type the existing Java Authentication and Authorization Services alias. The alias is used to retrieve the user name and password set on the configured J2C activation specification. The name is case sensitive and includes the node name.
6. In the **Service Operations** section, click **Edit Operations** to review the names of operations or add a description about the operations to be generated in the interface file.
7. 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. From the Discovered object list, 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**. By default, when the check box is not selected, field descriptions are used to generate properties. If you choose to use SAP field names to generate attribute names, two more check boxes are enabled:

b. Select the **Use SAP-original casing for Control Record Business object attribute names** check box to generate attribute names in SAP original casing. If unchecked, attribute names are generated using field names from SAP in camel-casing.

c. Select the **Use SAP-original casing for Data Record Business object attribute names** check box to generate attribute names in SAP original casing. If unchecked, attribute names are generated using field Name from SAP in camel-casing.

The following are the different possible combinations:

Table 18.

Scenario	Use SAP field names to generate attribute names (check-box)	Use SAP-original casing Control Record business object attribute names (check-box)	Use SAP-original casing Data Record business object attribute names (check-box)	Control Record	Data Record
1	Checked	Checked	Checked	Attribute names are generated using Field Name from SAP (Original SAP casing)	Attribute names are generated using Field Name from SAP (Original SAP casing)
2	Checked	Checked	Not checked	Attribute names are generated using Field Name from SAP (Original SAP casing)	Attribute names are generated using Field Name from SAP (Camel-casing)
3	Checked	Not checked	Checked	Attribute names are generated using Field Name from SAP (Camel-casing)	Attribute names are generated using Field Name from SAP (Original SAP casing)
4	Checked	Not checked	Not checked	Attribute names are generated using Field Name from SAP (Camel-casing)	Attribute names are generated using Field Name from SAP (Camel-casing)

Table 18. (continued)

Scenario	Use SAP field names to generate attribute names (check-box)	Use SAP-original casing Control Record business object attribute names (check-box)	Use SAP-original casing Data Record business object attribute names (check-box)	Control Record	Data Record
5	Not checked	Not checked	Not checked	Attribute names are generated using Field Description from SAP (Camel-casing)	Attribute names are generated using Field Description from SAP (Camel-casing)

d. 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.

e. 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.

   Note: Refer to Resolving data record (hexbinary) format issues for ALE inbound interfaces for more information on resolving data record format issues for ALE inbound interfaces.

f. 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. If the selected IDoc has unreleased segments, the **IDoc release version** property becomes required. It is recommended that you select the default value **Unreleased** if the IDoc you are working with has unreleased segments. If **Unreleased** is selected, the adapter generates the business objects for segments using the unreleased segment definition.

g. 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. The version of the SAP system is set as default. If the selected IDoc has unreleased segments, the **IDoc release version** property becomes required. It is recommended that you select **Unreleased** if the IDoc you are working with has unreleased segments. If **Unreleased** is selected, the adapter generates the business objects for segments using the unreleased segment definition.

h. 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.
- 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 .

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**. By default, when the check box is not selected, field descriptions are used to generate properties. If you choose to use SAP field names to generate attribute names, two more check boxes are enabled:

b. Select the **Use SAP-original casing for Control Record Business object attribute names** check box to generate attribute names in SAP original casing. If unchecked, attribute names are generated using field names from SAP in camel-casing.

c. Select the **Use SAP-original casing for Data Record Business object attribute names** check box to generate attribute names in SAP original casing. If unchecked, attribute names are generated using field Name from SAP in camel-casing.

The following are the different possible combinations:

Table 19.

Scenario	Use SAP field names to generate attribute names (check-box)	Use SAP-original casing Control Record business object attribute names (check-box)	Use SAP-original casing Data Record business object attribute names (check-box)	Control Record	Data Record
1	Checked	Checked	Checked	Attribute names are generated using Field Name from SAP (Original SAP casing)	Attribute names are generated using Field Name from SAP (Original SAP casing)
2	Checked	Checked	Not checked	Attribute names are generated using Field Name from SAP (Original SAP casing)	Attribute names are generated using Field Name from SAP (Camel-casing)
3	Checked	Not checked	Checked	Attribute names are generated using Field Name from SAP (Camel-casing)	Attribute names are generated using Field Name from SAP (Original SAP casing)
4	Checked	Not checked	Not checked	Attribute names are generated using Field Name from SAP (Camel-casing)	Attribute names are generated using Field Name from SAP (Camel-casing)

Table 19. (continued)

Scenario	Use SAP field names to generate attribute names (check-box)	Use SAP-original casing Control Record business object attribute names (check-box)	Use SAP-original casing Data Record business object attribute names (check-box)	Control Record	Data Record
5	Not checked	Not checked	Not checked	Attribute names are generated using Field Description from SAP (Camel-casing)	Attribute names are generated using Field Description from SAP (Camel-casing)

d. 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.

e. 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.

Note: Refer to Resolving data record (hexbinary) format issues for ALE inbound interfaces for more information on resolving data record format issues for ALE inbound interfaces.

f. 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.

g. 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 J2C Bean Creation 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 105.
- 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 artifactsSetting deployment properties and generating the service.

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

To configure the business object, use the following procedure.

Procedure

1. Select an IDoc to configure from the **IDocs selected** pane. You can select multiple IDocs to configure.
2. Select an operation (for example, **Create**) from the **Service operation** drop-down list.

   For each selected IDoc and receiving partner, you can configure the service operation. In addition to the default existing service operations which are `Create`, `Update` and `Delete`, you can specify a new operation by entering an operation name in the **Service operation** field.
3. Click **Add** to add the identifiers you want to associate with the operation. From the **IDoc Identifiers for the service operation:** list, select a set of identifiers to associate the Receiving partner, 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. You can associate multiple identifiers for a single service operation.

   Note: New operation will be added to the list only if you associate an identifier with the operation.
4. 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.

   If you enter an invalid namespace value, an error message is displayed as follows: **This property should be a URL value** and the next field becomes disabled. Valid namespace values are URLs that are valid as per the Java that supports it.
5. 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.

   Note: The above two fields cannot be edited if you are modifying existing artifacts.

   Note: If you are creating multiple adapter artifacts within a module, ensure that you specify different business object folders for each adapter within the module. For example, if you are creating artifacts for Oracle, JDBC, SAP, and JDE within a module, you need to create different relative folders for each of these adapters. If you do not specify different relative folders, the existing artifacts are overwritten when you generate new artifacts.
6. Click **Next**.

Results

You have associated an operation with an identifier. The J2C Bean Creation 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 the 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 EJB 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 required 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 required 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, select the JNDI name for an existing activation specification in WebSphere Application Server or create an activation specification. For more information about creating an activation

specification in WebSphere Application Server, see Setting activation specification properties for stand-alone adapters.

- To select an existing activation specification, click **Browse**.

Note: When you try to look up an existing JNDI in the JNDI Lookup Wizard screen, the wizard looks up only the JNDI that is created using the first RAR file. If you have more than one RAR file for the same adapter in the server, you can view only the JNDI generated using the first RAR file. The JNDI generated from the other RAR files are not looked up by the wizard.

- To create a new activation specification, click **New**.
  a. In the Server selection window, select the server on which the resource adapter will be deployed and click **Next**.
  b. In the New J2C Activation Specification window, type a JNDI name for the activation specification.
  c. To set additional properties, click **Advanced**.
     1) Select the **Use load balancing** check box to use load balancing to connect to the SAP system. Load balancing properties which are **Message server host**, **Logon group name** and **SAP system ID** have to be specified in the Additional connection configuration panel under the Advanced tab.
     2) The **Gateway host** is filled in, by default, with the value from the **Host name** field.
     3) A default value of **sapgw00** is filled in for **Gateway service**. If you have more than one gateway server in your SAP configuration, change **sapgw00** to the correct value.

        Note: See "Activation specification properties for BAPI inbound processing" on page 250 for more information.
     4) **Event polling configuration**
        a) In **Interval between polling periods**, type the number of milliseconds that the adapter waits between polling periods.
        b) In **Maximum events in polling period**, type the number of events to deliver in each polling period.
        c) In **Time between retries in case of system connection failure (milliseconds)**, specify the time interval between attempts to restart the event listeners.
        d) In **Maximum number of retries in case of system connection failure**, specify the number of times the adapter should try attempt to restart the event listeners.
        e) Select the **Stop the adapter when an error is encountered while polling** check box to specify whether the adapter should stop polling for events when it encounters an error during polling.
        f) Select the **Retry EIS connection on startup** check box to specify if you want the adapter to retry a failed connection when starting.
     5) **Event delivery configuration**
        a) Select **Type of delivery** to specify the order in which events are delivered by the adapter to the export.
        b) Select the **Ensure assured-once event delivery** check box to get a once-only delivery of the inbound events. This may reduce performance.

c) Select the **Do not process events that have a timestamp in the future** to specify whether the adapter should filter out future events by comparing the timestamp on each event with the system time.

d) Select the **Event types to process** to specify a list of event types that indicates to the adapter which events it should deliver.

e) Select the **Retry limit for failed events** to specify the number of times that the adapter attempts to re-deliver an event before marking the event as failed.

6) If you are using Secure Network Connection, select the **Secure Network Connection (SNC)** check box. Then enter information in the associated fields (name, partner, security level, and library path). Optionally type the name of an X509 certificate.

7) Optionally to set RFC and JCo tracing properties, perform the following steps:

a) Expand **SAP RFC trace configuration** and select **RFC trace on**.

b) Select the **Write JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API trace logged into the broker trace file will have the string `[JCoAPI]`. All JCo API traces are logged at the `INFO` message log level.

c) Select a tracing level from the **RFC trace level** list.

d) Click **Browse** and select a location to which the RFC trace files will be saved.

8) **AEP Data Format configuration**

a) Select **IDoc empty tags** to include empty tags to the unpopulated fields in the IDoc segment, which are sent to a configured endpoint, based on the option selected.

See Activation specification properties for more information about these properties.

d. Click **Finish** to return to the J2C Bean Creation and Deployment Configuration window.

5. Type the existing Java Authentication and Authorization Services alias. The alias is used to retrieve the user name and password set on the configured J2C activation specification. The name is case sensitive and includes the node name.
6. In the **Service Operations** section, click **Edit Operations** to review the names of operations or add a description about the operations to be generated in the interface file.
7. 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.

Note: The adapter now provides an out-of-the-box functionality to parse out the stream data provided in Passthrough IDoc Business Objects.

To use this new feature, you need to configure the DataBinding (after the EMD run) to use the `com.ibm.j2ca.sap.ale.idoc.datahandler.SAPIDocDataHandler` DataHandler Java class that is now bundled with the SAP adapter RAR file.

When invoking the Data Handler, you must provide the following details:

1. Name of the specific IDoc For example, `SapAlereq01`. The corresponding schema must be available for the Artifact Loader at runtime.
2. Business Object namespace for that specific schema.
3. Data Encoding.

With this setting, your end point will get the specific parsed IDoc without any change to the existing modules.

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 Properties window is displayed, you can set the following configurable options for the selected IDoc:
   a. Enable the checkbox **Parse the IDoc Control Record**, to generate a child business object to hold the parsed Control Record. When enabled, the property to send IDocs in the Flat File format are disabled. This feature enables you to route IDocs based on individual Control Record settings.
   b. Indicate whether you want to have multiple IDocs sent as one packet instead of sending them as individual business objects:
      - If you want to send multiple IDocs as a packet, select the **Send an IDoc packet as one business object** option.
   c. Indicate whether you want to send the IDoc in the Flat File format to the configured end point(s):

- If you do not want to send the IDoc in the Flat File format, leave the **Send IDoc in flat-file format** option unchecked
- If you want to send IDoc in the Flat File format, select the **Send IDoc in flat-file format** option:
  1) When sending the IDoc in the Flat File format, the Control Record length will remain 524. The Data Record length can vary depending on the segment length which is based on the IDoc version selected. Choose the correct value in the **IDoc release version** field. If the IDoc contains unreleased segments, leave the version field blank.
  2) When sending the IDoc in the Flat File format, set a delimiter, which will be added after the Control Record as well as after each Data Record, in the **Specify a delimiter for IDocs** field. You can choose standard delimiters provided in the drop down or specify a custom one.

4. Click **OK**. Then 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 the 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 EJB 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 theJ2C 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 required 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 required 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, select the JNDI name for an existing activation specification in WebSphere Application Server or create an activation specification. For more information about creating an activation specification in WebSphere Application Server, see Setting activation specification properties for stand-alone adapters.
   - To select an existing activation specification, click **Browse**.

     Note: When you try to look up an existing JNDI in the JNDI Lookup Wizard screen, the wizard looks up only the JNDI that is created using the first RAR file. If you have more than one RAR file for the same adapter in the server, you can view only the JNDI generated using the first RAR file. The JNDI generated from the other RAR files are not looked up by the wizard.
   - To create a new activation specification, click **New**.
     – In the Server selection window, select the server on which the resource adapter will be deployed and click **Next**.
     – In the New J2C Activation Specification window, type a JNDI name for the activation specification.
     – To set additional properties, click **Advanced**.
       a. Select the **Use load balancing** check box to use load balancing to connect to the SAP system. Load balancing properties which are **Message server host**, **Logon group name** and **SAP system ID** have to be specified in the Additional connection configuration panel under the Advanced tab.
       b. The **Gateway host** is filled in, by default, with the value from the **Host name** field.
       c. A default value of **sapgw00** is filled in for **Gateway service**. If you have more than one gateway server in your SAP configuration, change **sapgw00** to the correct value.

          Note: See “Activation specification properties for BAPI inbound processing” on page 250 for more information.
       d. Select **Ignore IDoc packet errors** if you want to continue to processing an IDoc packet if any errors occur during IDoc processing. If you want to provide update status for ALE processing, select **ALE update status** and fill in the associated fields. Properties marked with an (*) are required. Select **Send ALEAUD per packet** if you want to send one ALEAUD per IDoc packet which will contain confirmations for all IDocs in the packet.
       e. If you are using Secure Network Connection, select the **Secure Network Connection (SNC)** check box. Then enter information in the

associated fields (name, partner, security level, and library path). Optionally type the name of an X509 certificate.

f. Select the **Trim ALE IDoc field data** to specify if the leading white spaces are to be trimmed by the adapter before sending it to endpoint. If you want to include empty tags to the unpopulated fields in the IDoc segment, select the **Include IDoc empty tags** option.

g. To set RFC and JCo tracing properties, select the **RFC trace on** check box. Select the **Write SAP JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API trace logged into the broker trace file will have the string [JCoAPI]. All JCo API traces are logged at the INFO message log level.

5. Click **Finish**.

   See "Managed connection factory properties" on page 219 for more information about these properties.

   Properties marked with an asterisk (*) are required.

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 151.

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.

   If you enter an invalid namespace value, an error message is displayed as follows: **This property should be a URL value** and the next field becomes disabled. Valid namespace values are URLs that are valid as per the Java that supports it.
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.

   Note: The above two fields cannot be edited if you are modifying existing artifacts.

   Note: If you are creating multiple adapter artifacts within a module, ensure that you specify different business object folders for each adapter within the module. For example, if you are creating artifacts for Oracle, JDBC, SAP, and JDE within a module, you need to create different relative folders for each of these adapters. If you do not specify different relative folders, the existing artifacts are overwritten when you generate new artifacts.
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 the 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 EJB 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 required 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 required 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, select the JNDI name for an existing activation specification in WebSphere Application Server or create an activation specification. For more information about creating an activation specification in WebSphere Application Server, see Setting activation specification properties for stand-alone adapters.
   - To select an existing activation specification, click **Browse**.

   Note: When you try to look up an existing JNDI in the JNDI Lookup Wizard screen, the wizard looks up only the JNDI that is created using the first RAR file. If you have more than one RAR file for the same adapter in the server, you can view only the JNDI generated using the first RAR file. The JNDI generated from the other RAR files are not looked up by the wizard.
   - To create a new activation specification, click **New**.
     a. In the Server selection window, select the server on which the resource adapter will be deployed and click **Next**.
     b. In the New J2C Activation Specification window, type a JNDI name for the activation specification.

c. To set additional properties, click **Advanced**.
   1) Select the **Use load balancing** check box to use load balancing to connect to the SAP system. Load balancing properties which are **Message server host**, **Logon group name** and **SAP system ID** have to be specified in the Additional connection configuration panel under the Advanced tab.
   2) The **Gateway host** is filled in, by default, with the value from the **Host name** field.
   3) A default value of **sapgw00** is filled in for **Gateway service**. If you have more than one gateway server in your SAP configuration, change **sapgw00** to the correct value.

      Note: See "Activation specification properties for BAPI inbound processing" on page 250 for more information.
   4) **Event polling configuration**
      a) In **Interval between polling periods**, type the number of milliseconds that the adapter waits between polling periods.
      b) In **Maximum events in polling period**, type the number of events to deliver in each polling period.
      c) In **Time between retries in case of system connection failure (milliseconds)**, specify the time interval between attempts to restart the event listeners.
      d) In **Maximum number of retries in case of system connection failure**, specify the number of times the adapter should try attempt to restart the event listeners.
      e) Select the **Stop the adapter when an error is encountered while polling** check box to specify whether the adapter should stop polling for events when it encounters an error during polling.
      f) Select the **Retry EIS connection on startup** check box to specify if you want the adapter to retry a failed connection when starting.
   5) **Event delivery configuration**
      a) Select **Type of delivery** to specify the order in which events are delivered by the adapter to the export.
      b) Select the **Ensure assured-once event delivery** check box to get a once-only delivery of the inbound events. This may reduce performance.
      c) Select the **Do not process events that have a timestamp in the future** to specify whether the adapter should filter out future events by comparing the timestamp on each event with the system time.
      d) Select the **Event types to process** to specify a list of event types that indicates to the adapter which events it should deliver.
      e) Select the **Retry limit for failed events** to specify the number of times that the adapter attempts to re-deliver an event before marking the event as failed.
   6) If you are using Secure Network Connection, select the **Secure Network Connection (SNC)** check box. Then enter information in the associated fields (name, partner, security level, and library path). Optionally type the name of an X509 certificate.
   7) Optionally to set RFC and JCo tracing properties, perform the following steps:

a) Expand **SAP RFC trace configuration** and select **RFC trace on**.
b) Select the **Write JCo traces into adapter logs** checkbox to enable JCo API traces to be generated in the broker trace file. These traces will then be interleaved with the adapter traces. Each JCo API trace logged into the broker trace file will have the string `[JCoAPI]`. All JCo API traces are logged at the `INFO` message log level.
c) Select a tracing level from the **RFC trace level** list.
d) Click **Browse** and select a location to which the RFC trace files will be saved.

8) **AEP Data Format configuration**
a) Select **IDoc empty tags** to include empty tags to the unpopulated fields in the IDoc segment, which are sent to a configured endpoint, based on the option selected.

See Activation specification properties for more information about these properties.

d. Click **Finish** to return to the J2C Bean Creation and Deployment Configuration window.

5. Type the existing Java Authentication and Authorization Services alias. The alias is used to retrieve the user name and password set on the configured J2C activation specification. The name is case sensitive and includes the node name.
6. In the **Service Operations** section, click **Edit Operations** to review the names of operations or add a description about the operations to be generated in the interface file.
7. Click **Finish**.

Chapter 5. Deploying the module

Deploy a module to place the files that make up your module and adapter into an operational environment for testing. In Rational Application Developer for WebSphere Software, the integrated test environment features runtime support for WebSphere Application Server, depending on the test environment profiles that you selected during installation.

Deployment environments

There are test and production environments into which you can deploy modules and adapters.

In Rational Application Developer for WebSphere Software, you can deploy your modules to one or more servers in the test environment. This is typically the most common practice for running and testing projects. However, you can also export modules for server deployment on WebSphere Application Server as EAR files using the administrative console or command-line tools.

Deploying the module for testing

In Rational Application Developer for WebSphere Software, you can deploy a module that includes an embedded adapter to the test environment and work with server tools that enable you to perform such tasks as editing server configurations, starting, and stopping servers and testing the module code for errors. The testing is generally performed on the interface operations of your components, which enables you to determine whether the components are correctly implemented and the references are correctly wired.

Configuring the connector dependencies

Dependent JARs have to be added to the libraries directory or packaged in the EAR.

About this task

The JARs are set in the class path and these dependent libraries have to be made available for run time when the module is deployed. There are two ways to make the dependent libraries available, one for either stand-alone deployment or embedded deployment and the other for embedded deployment only.

Configuring the connector dependencies on the server

You must copy the required sapjco3.jar jar file and related files to your runtime environment before you can run your adapter applications.

About this task

To obtain the required files and copy them to WebSphere Application Server, use the following procedure.

1. Obtain the sapjco3.jar file and the associated files for your operating system from your SAP administrator or from the SAP Web site. Also obtain the

CWYAP_SAPAdapterExt.jar in the <IID_INSTALL_ROOT>>/ ResourceAdapters/SAP_7.0.0.0/ext folder from the adapter. The files are listed in the following table.

Note: The software dependencies differ, depending on the version of SAP Software Tools you use.

Table 20. External software dependency files required by SAP Software

Operating system	Files to be copied
Windows and i5/OS	sapjco3.jar, sapidoc3.jar, and any *.dll files that come with the SAP JCo download from the SAP Web site. CWYAP_SAPAdapterExt.jar from the SAP adapter.
UNIX (including UNIX System Services on z/OS)	sapjco3.jar, sapidoc3.jar and any .so and .o files that come with the SAP JCo download from the SAP Web site. CWYAP_SAPAdapterExt.jar from the SAP adapter.

2. For SAP Releases 6.40 and 7.0, unzip the R3DLLINST.ZIP archive (C runtime 7.1) from the SAP note 684106 and run the"R3DLLINS.EXE" file in the subdirectory NTPATCH. For SAP releases 4.6D EX2, Web AS 6.40 EX2, SAP NetWeaver 7.01 and 7.10 and higher, download the installation program **vcredist_<platform>.exe.** Then process the program. The vcredist_<platform > installation packages are delivered with the installation master DVDs of SAP releases 7.01 and 7.10 and are located in NPATCH directory.
3. SAP JCo requires **dbghelp.dll** on Windows environment. This dll is found in the system32 directory on most Windows systems. Copy this dll onto your Windows environment if you do not have it.
4. Copy the files to the server.
   - In a testing environment in Rational Application Developer for WebSphere Software, copy the files to the appropriate directory such as ${WAS_INSTALL_ROOT}/runtimes/bi_v62/lib/ext directory.
   - In a production environment, copy the files to the ${WAS_INSTALL_ROOT}/ lib/ext directory of WebSphere Application Server.
   - For z/OS, add the specified files to the following locations:
     a. Add the sapjco3.jar file, sapidoc3.jar file, and the CWYAP_SAPAdapterExt.jar file to the ${*WAS_INSTALL_ROOTWPS_INSTALL_ROOT*}/classes directory.
     b. Add the .so files to the ${*WAS_INSTALL_ROOT*}/lib directory.
   - For IBM i system, add the specified files and variables to the /SAPJCO directory:
     a. Add sapjco3.jar, sapidoc3.jar file, and any *.dll files that come with the SAP JCo download from the SAP Web site.
     b. Add the CWYAP_SAPAdapterExt.jar from the SAP adapter.
     c. Set the following at the *SYS level on the IBMi server:
        – Add a **LIBPATH** variable for the SAPJCO folder
        – Add a **CLASSPATH** variable that points to the `/SAPJCO/sapjco3.jar` directory
        – Add a system-wide variable **QIBM_JAVA_PASE_STARTUP** that points to the `/usr/lib/start64` directory
     d. Restart the server and deploy the application again
   - For all other operating systems, add the specified files to the following locations:

a. Add the SAP Java Connector interface (sapjco3.jar, sapidoc3.jar, and CWYAP_SAPAdapterExt.jar) to the lib subdirectory of the WebSphere Application Server.
b. Add the other SAP JCO files to the bin subdirectory of the WebSphere Application Server installation directory.

   The installation directory is typically in the runtimes\bi_v6 directory of the Rational Application Developer for WebSphere Software installation directory.

Results

The `sapjco3.jar` and associated files are now part of your runtime environment.

Configuring the connector dependencies when the adapter is bundled

You must copy the dependent JAR files to the EAR application before you can run your adapter applications. You must use this method only for embedded deployment.

About this task

To obtain the required files and copy them to the EAR application, use the following procedure:

Procedure

1. From the appropriate module, go to the workspace and copy the JAR files to the directory. For example if the name of the module is ModuleName, then go to the workspace and copy the JAR files to the `ModuleNameApp/EarContent` directory.
2. Modify the adapter RAR's manifest file, `manifest.mf`, with the list of JAR files required by the adapter. Add the JAR files in the following format: Class-Path: dependantjar1.jar, dependantjar2.jar.
3. Copy the native libraries to the run time bin directory and deploy the application.

Results

The vendor software libraries are now a part of your run time environment.

Adding the module to the server

In Rational Application Developer for WebSphere Software, you can add modules to one or more servers in the test environment.

Before you begin

If the module you are testing uses an adapter to perform inbound processing, generate and wire a *target component* to which the adapter sends the events.

About this task

In order to test your module and its use of the adapter, you need to add the module to the server.

Procedure

1. *Conditional:* If there are no servers in the **Servers** view, add and define a new server by performing the following steps:
   a. Place your cursor in the **Servers** view, right-click, and select **New** > **Server**.
   b. From the Define a New Server window, select the server type.
   c. Configure servers settings.
   d. Click **Finish** to publish the server.
2. Add the module to the server.
   a. Switch to the servers view. In Rational Application Developer for WebSphere Software, select **Windows** > **Show View** > **Servers**.
   a. Start the server. In the **Servers** tab in the lower-right pane of the Rational Application Developer for WebSphere Software screen, right-click the server, and then select **Start**.
3. When the server status is *Started*, right-click the server, and select **Add and Remove** .
4. In the Add and Remove Projects screen, select your project and click **Add**. The project moves from the **Available projects** list to the **Configured projects** list.
5. Click **Finish**. This deploys the module on the server.

   The Console tab in the lower-right pane displays a log while the module is being added to the server.

What to do next

Test the functionality of your module and the adapter. For information about testing a module using the test client, see the *Testing applications on a server* topic in the Rational Application Developer for WebSphere Software information center.

Deploying the module for production

Deploying a module created with the J2C Bean wizard to WebSphere Application Server in a production environment is a two-step process. First, you export the module in Rational Application Developer for WebSphere Software as an enterprise archive (EAR) file. Second, you deploy the EAR file using the WebSphere Application Server administrative console.

Configuring the connector dependencies on the server

You must copy the required sapjco3.jar jar file and related files to your runtime environment before you can run your adapter applications.

About this task

To obtain the required files and copy them to WebSphere Application Server, use the following procedure.

1. Obtain the sapjco3.jar file and the associated files for your operating system from your SAP administrator or from the SAP Web site. Also obtain the CWYAP_SAPAdapterExt.jar in the <IID_INSTALL_ROOT>>/ ResourceAdapters/SAP_7.0.0.0/ext folder from the adapter. The files are listed in the following table.

   Note: The software dependencies differ, depending on the version of SAP Software Tools you use.

Table 21. External software dependency files required by SAP Software

Operating system	Files to be copied
Windows and i5/OS	sapjco3.jar, sapidoc3.jar, and any *.dll files that come with the SAP JCo download from the SAP Web site. CWYAP_SAPAdapterExt.jar from the SAP adapter.
UNIX (including UNIX System Services on z/OS)	sapjco3.jar, sapidoc3.jar and any .so and .o files that come with the SAP JCo download from the SAP Web site. CWYAP_SAPAdapterExt.jar from the SAP adapter.

2. For SAP Releases 6.40 and 7.0, unzip the R3DLLINST.ZIP archive (C runtime 7.1) from the SAP note 684106 and run the"R3DLLINS.EXE" file in the subdirectory NTPATCH. For SAP releases 4.6D EX2, Web AS 6.40 EX2, SAP NetWeaver 7.01 and 7.10 and higher, download the installation program **vcredist_<platform>.exe.** Then process the program. The vcredist_<platform > installation packages are delivered with the installation master DVDs of SAP releases 7.01 and 7.10 and are located in NPATCH directory.
3. SAP JCo requires **dbghelp.dll** on Windows environment. This dll is found in the system32 directory on most Windows systems. Copy this dll onto your Windows environment if you do not have it.
4. Copy the files to the server.
   - In a testing environment in Rational Application Developer for WebSphere Software, copy the files to the appropriate directory such as ${WAS_INSTALL_ROOT}/runtimes/bi_v62/lib/ext directory.
   - In a production environment, copy the files to the ${WAS_INSTALL_ROOT}/ lib/ext directory of WebSphere Application Server.
   - For z/OS, add the specified files to the following locations:
     a. Add the sapjco3.jar file, sapidoc3.jar file, and the CWYAP_SAPAdapterExt.jar file to the ${*WAS_INSTALL_ROOTWPS_INSTALL_ROOT*}/classes directory.
     b. Add the .so files to the ${*WAS_INSTALL_ROOT*}/lib directory.
   - For IBM i system, add the specified files and variables to the /SAPJCO directory:
     a. Add sapjco3.jar, sapidoc3.jar file, and any *.dll files that come with the SAP JCo download from the SAP Web site.
     b. Add the CWYAP_SAPAdapterExt.jar from the SAP adapter.
     c. Set the following at the *SYS level on the IBMi server:
        – Add a **LIBPATH** variable for the SAPJCO folder
        – Add a **CLASSPATH** variable that points to the `/SAPJCO/sapjco3.jar` directory
        – Add a system-wide variable **QIBM_JAVA_PASE_STARTUP** that points to the `/usr/lib/start64` directory
     d. Restart the server and deploy the application again
   - For all other operating systems, add the specified files to the following locations:
     a. Add the SAP Java Connector interface (sapjco3.jar, sapidoc3.jar, and CWYAP_SAPAdapterExt.jar) to the lib subdirectory of the WebSphere Application Server.
     b. Add the other SAP JCO files to the bin subdirectory of the WebSphere Application Server installation directory.

The installation directory is typically in the runtimes\bi_v6 directory of the Rational Application Developer for WebSphere Software installation directory.

Results

The `sapjco3.jar` and associated files are now part of your runtime environment.

Installing the RAR file (for modules using stand-alone adapters only)

If you chose not to embed the adapter with your module, but instead choose to make the adapter available to all deployed applications in the server instance, you need to install the adapter in the form of a RAR file to the application server. A RAR file is a Java archive (JAR) file that is used to package a resource adapter for the Java 2 Connector (J2C) architecture.

Before you begin

You must set **Deploy connector project** to **On server for use by multiple adapters** in the J2C Bean Creation and Deployment Configuration window of the J2C Bean wizard.

About this task

Installing the adapter in the form of a RAR file results in the adapter being available to all Java EE application components running in the server run time.

Procedure

1. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration** > **Run administrative console**.
3. Log on to the administrative console.
4. Click **Resources** > **Resource Adapters** > **Resource adapters**.
5. In the Resource adapters page, click **Install RAR**.
6. In the Install RAR file page, click **Browse** and navigate to the RAR file for your adapter.

   The RAR files are typically installed in the following path:
   RAD_installation_directory/ResourceAdapters/*adapter_name*/*adapter*.rar
7. Click **Next**.
8. Optional: In the Resource adapters page, change the name of the adapter and add a description.
9. Click **OK**.
10. Click **Save** in the **Messages** box at the top of the page.

What to do next

The next step is to export the module as an EAR file that you can deploy on the server.

Exporting the module as an EAR file

Using Rational Application Developer for WebSphere Software, export your module as an EAR file. By creating an EAR file, you capture all of the contents of your module in a format that can be easily deployed to WebSphere Application Server.

Before you begin

Before you can export a module as an EAR file, you must have created a module to communicate with your service. The module should be displayed in the Rational Application Developer for WebSphere Software Enterprise Explorer view.

About this task

To export the module as an EAR file, perform the following procedure.

Procedure

1. Right-click the module and select **Export**.
2. In the Select window, expand **Java EE**.
3. Select **EAR file** and click **Next**.
4. Optional: Select the correct EAR application. The EAR application is named after your module, but with "App" added to the end of the name.
5. Browse for the folder on the local file system where the EAR file will be placed.
6. Optional: To export the source files, select the **Export source files** check box. This option is provided in case you want to export the source files in addition to the EAR file. Source files include files associated with Java components, data maps, and so on.
7. Optional: To overwrite an existing file, click **Overwrite existing file**.
8. Click **Finish**.

Results

The contents of the module are exported as an EAR file.

What to do next

Install the module in the administrative console. This deploys the module to WebSphere Application Server.

Installing the EAR file

Installing the EAR file is the last step of the deployment process. When you install the EAR file on the server and run it, the adapter, which is embedded as part of the EAR file, runs as part of the installed application.

Before you begin

You must have exported your module as an EAR file before you can install it on WebSphere Application Server.

About this task

To install the EAR file, perform the following procedure. For more information about clustering adapter module applications, see the http://www.ibm.com/

software/webservers/appserv/was/library/.

Procedure

1. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration** > **Run administrative console**.
3. Log on to the administrative console.
4. Click **Applications** > **New Application** > **New Enterprise Application**.
5. Click **Browse** to locate your EAR file and click **Next**. The EAR file name is the name of the module followed by "App."
6. Optional: If you are deploying to a clustered environment, complete the following steps.
   a. On the **Step 2: Map modules to servers** window, select the module and click **Next**.
   b. Select the name of the server cluster.
   c. Click **Apply**.
7. Click **Next**. In the Summary page, verify the settings and click **Finish**.
8. Optional: If you are using an authentication alias, complete the following steps:
   a. Expand **Security** and select **Business Integration Security**.
   b. Select the authentication alias that you want to configure. You must have administrator or operator rights to change the authentication alias configurations.
   c. Optional: If it is not already specified, type the **User name**.
   d. If it is not already specified, type the **Password**.
   e. If it is not already specified, type the password again in the **Confirm Password** field.
   f. Click **OK**.

Results

The project is now deployed and the Enterprise Applications window is displayed.

What to do next

If you want to set or reset any properties or you would like to cluster adapter project applications, make those changes using the administrative console before configuring troubleshooting tools.

Deploying the module in a clustered environment

In Rational Application Developer for WebSphere Software, you can deploy the IBM WebSphere Adapter for SAP Software in a clustered environment.

To deploy the module in a clustered environment, use any of the following approaches.

- **Embedded module:** The adapter is embedded in the application and specific to it. The adapter cannot be shared between multiple applications.
- **Node level module with embedded activation specification:** The adapter is deployed at the node level, with the activation specification created during module creation. The adapter can be shared across multiple applications.

- **Node level module with JNDI activation specification reference:** The adapter is deployed at the node level, and the application provides a JNDI reference to the activation specification. You must create the reference at the cluster scope from the administrative console, with the same JNDI name. The adapter can be shared across multiple applications.

Deploying module embedded in the application

The adapter is deployed embedded in the application and specific to it. The adapter cannot be shared between multiple applications.

Before you begin

The following steps are a necessary prerequisite to configure and deploy the module.

- Rational Application Developer for WebSphere Software version 7.5.0.0 or above.
- A clustered topology deployment environment on the WebSphere Application Server available from Rational Application Developer for WebSphere Software.
- Create a clustered topology deployment environment, as shown in the following **Gold Topology** configuration figure.
- Deploy the adapter and the adapter applications (EAR files) in the AppTarget (the target that hosts the SCA container).

About this task

To create an application with the embedded adapter, use the J2C Bean wizard.

Procedure

1. In the Service Configuration Properties window, from the **Deploy connector project** property list, select **With module for use by single application**.
2. Create the module as described in the Business process management samples for WebSphere Adapters.
3. In the **Dependencies** option for the module, after the module is created, ensure that the **Deploy with module** option is selected for the adapter.
4. If the server is not running, right-click your server in the **Servers** view and select **Start**.
5. When the server status changes to **Started**, right-click the server and select **Administration** > **Run administrative console**.
6. From the **Deployment Manager Admin Console**, click **Install applications** to deploy the application.
7. On the **Step 2: Map modules to servers** window, select the module and click **Next**. For the embedded adapter option, the adapter is deployed as part of the application.
8. In the Enterprise Applications view, select the new application **<*adapter_name*>EmbeddedModuleApp**. The new application is displayed after the application is deployed at the deployment manger level.
9. Select the node and click **Installed applications** to view the deployed application on each individual node.

Results

The resource adapter is embedded and deployed as part of the application.

Deploying module at node level with embedded activation specification

The adapter is deployed at the node level, with the activation specification created during module creation. The adapter can be shared across multiple applications.

Before you begin

The following steps are a necessary prerequisite to configure and deploy the module.

- Rational Application Developer for WebSphere Software version 7.5.0.0 or above.
- A clustered topology deployment environment on the WebSphere Application Server available from Rational Application Developer for WebSphere Software.
- Create a clustered topology deployment environment, as shown in the following **Gold Topology** configuration figure.
- Deploy the adapter and the adapter applications (EAR files) in the AppTarget (the target that hosts the SCA container).

About this task

To create an application with the node level adapter and activation specification properties specified in the module itself, use the J2C Bean wizard.

Procedure

1. In the Service Configuration Properties window, from the **Deploy connector project** property list, select **On server for use by multiple applications**.
2. From the **Connection properties** list, select **Use properties below**.
3. Create the module as described in the Business process management samples for WebSphere Adapters.
4. In the **Dependencies** option for the module, ensure that the **Deploy with module** option is not selected for the adapter. Here, the adapter is not part of the module, therefore you must deploy the adapter before deploying the application.
5. If the server is not running, right-click your server in the **Servers** view and select **Start**.
6. When the server status changes to **Started**, right-click the server and select **Administration** > **Run administrative console**. Log on to the administrative console.
7. To deploy the adapter at individual nodes, click **Resources** > **Resource Adapters** > **Resource adapters**. In the clustered environment, you must install the adapter in each node separately.
8. In the Resource adapters page, click **Install RAR**.
9. In the Install RAR file page, click **Browse** and navigate to the RAR file for your adapter. Deploy the RAR on each node.

   The RAR files are typically installed in the following path:
   RAD_installation_directory/ResourceAdapters/*adapter_name*/*adapter*.rar
10. For deployment at node level, do not select any **Scope** because the scope is always **Node**. Click **Next**.
11. Optional: In the Resource adapters page, change the name of the adapter and add a description. Click **OK**.
12. Click **Save** in the **Messages** box at the top of the page.

13. For node level deployment, check if the adapter RAR is deployed at the node level.
14. To deploy the adapter at the cluster level, click **Resources** > **Resource Adapters** > **Resource adapters**.
15. In the Resource adapters window, set the **Scope** to **Cluster**, and then click **New**.
16. Select the RAR deployed at the node level.
17. Check if the adapter RAR is now deployed at the cluster level. Deploy the application after the adapter is deployed at the node level on the individual nodes, and then at the cluster level.
18. From the **Deployment Manager Admin Console**, click **Install applications** to deploy the application.
19. On the **Step 2: Map modules to servers** window, select the module and click **Next**. The adapter is not part of the deployed application.
20. In the **Admin Console**, click **Resources** > **Resource Adapters** > **IBM WebSphere Adapter for SAP Software** > **J2C activation specifications** to view the activation specification from the adapter deployed at the cluster level.

Results

The resource adapter is deployed at the node level, with the activation specification.

Deploying module at node level with JNDI activation specification

The adapter is deployed at the node level, and the application provides a JNDI reference to the activation specification. You must create the activation specification with the same JNDI name at the cluster scope from the administrative console. The adapter can be shared across multiple applications

Before you begin

The following steps are a necessary prerequisite to configure and deploy the module.

- Rational Application Developer for WebSphere Software version 7.5.0.0 or above.
- A clustered topology deployment environment on the WebSphere Application Server available from Rational Application Developer for WebSphere Software.
- Create a clustered topology deployment environment, as shown in the following **Gold Topology** configuration figure.
- Deploy the adapter and the adapter applications (EAR files) in the AppTarget (the target that hosts the SCA container).

About this task

To create an application with the node level adapter and activation specification properties specified in the module itself, use the J2C Bean wizard.

Procedure

1. In the Service Configuration Properties window, from the **Deploy connector project** property list, select **On server for use by multiple applications**.

2. From the **Connection properties** list, select **Use JNDI lookup name configured on server**.
3. In the **JNDI lookup name** property field, specify the JNDI name. Use this same JNDI name when you create the activation specification from the Admin Console.
4. Create the module as described in the Business process management samples for WebSphere Adapters.
5. In the **Dependencies** option for the module, ensure that the **Deploy with module** option is not selected for the adapter.
6. If the server is not running, right-click your server in the **Servers** view and select **Start**.
7. When the server status changes to **Started**, right-click the server and select **Administration** > **Run administrative console**. Log on to the administrative console.
8. To install the adapter at the node level, click **Resources** > **Resource Adapters** > **Resource adapters**. In the clustered environment, you must install the adapter in each node separately.
9. In the Resource adapters page, click **Install RAR**.
10. In the Install RAR file page, click **Browse** and navigate to the RAR file for your adapter. Deploy the RAR on each node.

    The RAR files are typically installed in the following path:
    IID_installation_directory/ResourceAdapters/*adapter_name*/*adapter*.rar
11. For deployment at node level, do not select any **Scope** because the scope is always **Node**. Click **Next**.
12. Optional: In the Resource adapters page, change the name of the adapter and add a description. Click **OK**.
13. Click **Save** in the **Messages** box at the top of the page.
14. To install the RAR at the cluster level, click **Resources** > **Resource Adapters** > **Resource adapters**
15. In the Resource adapters page, set the **Scope** to **Cluster**, and then click **New**.
16. Select the RAR deployed at the node level, and then check if the adapter RAR is now deployed at the cluster level. Deploy the application after the adapter is deployed at the node level on the individual nodes, and then at the cluster level.
17. From the **Deployment Manager Admin Console**, click **Install applications** to deploy the application.
18. In the **Admin Console**, click **Resources** > **Resource Adapters** > **IBM WebSphere Adapter for SAP Software** > **J2C activation specifications** > **New** to create the activation specification from the adapter deployed at the cluster level.
19. When installing the adapter, in the **Name** field, you must enter the same name as defined in the RAR.
20. In the **JNDI name** field, you must enter the same name as given during the module creation.
21. Click **Resources** > **Resource Adapters** > **IBM WebSphere Adapter for SAP Software** > **J2C activation specifications** to check if the JNDI reference on the adapter is same as the one specified for the module.
22. Click **Resources** > **Resource Adapters** > **IBM WebSphere Adapter for SAP Software** > **J2C activation specifications** > **Custom properties** to set values for the activation specification in the Admin Console.

23. From the **Deployment Manager Admin console**, click **Install applications** to deploy the application after you deploy the RAR and create the activation specification.
24. On the **Step 2: Map modules to servers** page, select the module and click **Next**. The adapter is not part of the deployed application.

Results

The resource adapter is deployed at the node level, with the JNDI activation specification reference.

Chapter 6. Troubleshooting and support

Common troubleshooting techniques and self-help information help you identify and solve problems quickly. For information about configuring logging properties and changing the log and trace file names, see Configuring logging and tracing.

Techniques for troubleshooting problems

Troubleshooting is a systematic approach to solving a problem. The goal is to determine why something does not work as expected and how to resolve the problem. Certain common techniques can help with the task of troubleshooting.

The first step in the troubleshooting process is to describe the problem completely. Without a problem description, neither you or IBM® can know where to start to find the cause of the problem. This step includes asking yourself basic questions, such as:

- What are the symptoms of the problem?
- Where does the problem occur?
- When does the problem occur?
- Under which conditions does the problem occur?
- Can the problem be reproduced?

The answers to these questions typically lead to a good description of the problem, and that is the best way to start down the path of problem resolution.

What are the symptoms of the problem?

When starting to describe a problem, the most obvious question is "What is the problem?" Which might seem like a straightforward question; however, you can break it down into several more-focused questions that create a more descriptive picture of the problem. These questions can include:

- Who, or what, is reporting the problem?
- What are the error codes and messages?
- How does the system fail? For example, is it a loop, hang, lock up, performance degradation, or incorrect result?
- What is the business impact of the problem?

Where does the problem occur?

Determining where the problem originates is not always simple, but it is one of the most important steps in resolving a problem. Many layers of technology can exist between the reporting and failing components. Networks, disks, and drivers are only a few components to be considered when you are investigating problems.

The following questions can help you to focus on where the problem occurs in order to isolate the problem layer.

- Is the problem specific to one platform or operating system, or is it common for multiple platforms or operating systems?
- Is the current environment and configuration supported?

Remember that if one layer reports the problem, the problem does not necessarily originate in that layer. Part of identifying where a problem originates is understanding the environment in which it exists. Take some time to completely

describe the problem environment, including the operating system and version, all corresponding software and versions, and hardware information. Confirm that you are running within an environment that is a supported configuration; many problems can be traced back to incompatible levels of software that are not intended to run together or have not been fully tested together.

When does the problem occur?

Develop a detailed timeline of events leading up to a failure, especially for those cases that are one-time occurrences. You can most simply do this by working backward: Start at the time an error was reported (as precisely as possible, even down to the millisecond), and work backward through the available logs and information. Typically, you need to look only as far as the first suspicious event that you find in a diagnostic log; however, this is not always simple to do and takes practice. Knowing when to stop looking is especially difficult when multiple layers of technology are involved, and when each has its own diagnostic information.

To develop a detailed timeline of events, answer the following questions:
- Does the problem happen only at a certain time of day or night?
- How often does the problem happen?
- What sequence of events leads up to the time that the problem is reported?
- Does the problem happen after an environment change, such as upgrading or installing software or hardware?

Responding to these types of questions can provide you with a frame of reference in which to investigate the problem.

Under which conditions does the problem occur?

Knowing what other systems and applications are running at the time that a problem occurs is an important part of troubleshooting. These and other questions about your environment can help you to identify the root cause of the problem:
- Does the problem always occur when the same task is being performed?
- Does a certain sequence of events need to occur for the problem to surface?
- Do any other applications fail at the same time?

Answering these types of questions can help you explain the environment in which the problem occurs and correlate any dependencies. Remember that just because multiple problems might have occurred around the same time, the problems are not necessarily related.

Can the problem be reproduced?

From a troubleshooting standpoint, the "ideal" problem is one that can be reproduced. Typically with problems that can be reproduced, you have a larger set of tools or procedures at your disposal to help you investigate. Consequently, problems that you can reproduce are often simpler to debug and solve. However, problems that you can reproduce can have a disadvantage: If the problem is of significant business impact, you do not want it to recur! If possible, re-create the problem in a test or development environment, which typically offers you more flexibility and control during your investigation.

Tip: Simplify the scenario to isolate the problem to a suspected component.

The following questions can help you with reproducing the problem:

- Can the problem be re-created on a test machine?
- Are multiple users or applications encountering the same type of problem?
- Can the problem be re-created by running a single command, a set of commands, a particular application, or a stand-alone application?

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, ensure that the first eight characters of the adapter ID property are unique for each instance so that you can correlate the 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 the log and trace information to a particular instance of an adapter. For example, when 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. If you set the adapter ID properties of two instances to `Instance01` and `Instance02`, you will not be able to 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. It prevents 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, see the “Adapter ID (AdapterID)” on page 217 property.

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

To avoid data being wrongly truncated with a delimiter, you must change the default function module used by the adapter to retrieve data from SAP tables in the query interface for SAP Software, which processes non-unicode systems for other than English languages.

Before you begin

You must have developer access to SAP system and you should have basic knowledge of advanced business application programming (ABAP) to create a custom module.

About this task

On non-unicode systems, the default function used to retrieve data from SAP tables (RFC_READ_TABLE) might truncate the data wrongly. To avoid the data being wrongly truncated, create a custom function on the SAP server and use this newly created function to retrieve data.

Use the following procedure to create the custom retrieve function and to specify it during configuration:

Procedure

1. Enter an appropriate delimiter in the XSD schema file. In the case of KNA1, the schema generated is SapKna1.xsd.
2. Open this file in a text editor. Ensure that the entered delimiter is unique. By default, the value is set to the pipe symbol (|).
3. In the SAP EIS, Go to SE37 tcode, click **Goto** > **Function Groups** > **Create Group**. The Create Function Group window is displayed.
4. In the **Function Group** field, type ZRFC_READ_TABLE as the function group name.
5. In the **Short text** field, type a short description for the function group. For example, type function Group for ZRFC_READ_TABLE as short text for the function group.
6. In the **Person Responsible** field, type the name of the person responsible for creating the function group. Click **Save**.
7. Go to SE80 tcode to activate the function group.
8. Select the package name where you have saved the ZRFC_READ_TABLE.
9. Click the display button and expand **Function Groups**.
10. Select ZRFC_READ_TABLE , select **Activate** using right-click option.
11. Go to SE37 tcode to copy the function group.
12. In the **Function Module** field, enter ZRFC_READ_TABLE.
13. Click **Function Module** > **Other Functions** > **Copy**. The Copy Function Module window is displayed.
14. In the **fr. Function module** field, type RFC_READ_TABLE.
15. In the **To Function module** field, type ZRFC_READ_TABLE.
16. In the **Function group** field, type ZRFC_READ_TABLE. Click **Copy**. Click **Change** in the SE37 tcode window.
17. Select **Source code** tab and copy the following code:

```
FUNCTION ZRFC_READ_TABLE.
*"----------------------------------------------------------------------
*"*"Local interface:
*"  IMPORTING
*"     VALUE(QUERY_TABLE) LIKE  DD02L-TABNAME
*"     VALUE(DELIMITER) LIKE  SONV-FLAG DEFAULT SPACE
*"     VALUE(NO_DATA) LIKE  SONV-FLAG DEFAULT SPACE
*"     VALUE(ROWSKIPS) LIKE  SOID-ACCNT DEFAULT 0
*"     VALUE(ROWCOUNT) LIKE  SOID-ACCNT DEFAULT 0
*"  TABLES
*"      OPTIONS STRUCTURE  RFC_DB_OPT
*"      FIELDS STRUCTURE  RFC_DB_FLD
*"      DATA STRUCTURE  TAB512
*"      WA1 STRUCTURE  TAB512
*"  EXCEPTIONS
*"      TABLE_NOT_AVAILABLE
*"      TABLE_WITHOUT_DATA
*"      OPTION_NOT_VALID
*"      FIELD_NOT_VALID
*"      NOT_AUTHORIZED
*"      DATA_BUFFER_EXCEEDED
*"----------------------------------------------------------------------
"
CALL FUNCTION 'VIEW_AUTHORITY_CHECK'
     EXPORTING
          VIEW_ACTION                    = 'S'
          VIEW_NAME                      = QUERY_TABLE
     EXCEPTIONS
          NO_AUTHORITY                   = 2
```

```
          NO_CLIENTINDEPENDENT_AUTHORITY = 2
          NO_LINEDEPENDENT_AUTHORITY     = 2
          OTHERS                         = 1.

IF SY-SUBRC = 2.
  RAISE NOT_AUTHORIZED.
ELSEIF SY-SUBRC = 1.
  RAISE TABLE_NOT_AVAILABLE.
ENDIF.

* ---------------------------------------------------------------------
*  find out about the structure of QUERY_TABLE
* ---------------------------------------------------------------------
DATA BEGIN OF TABLE_STRUCTURE OCCURS 10.
        INCLUDE STRUCTURE DFIES.
DATA END OF TABLE_STRUCTURE.
"DATA TABLE_HEADER LIKE X030L.
DATA TABLE_TYPE TYPE DD02V-TABCLASS.

CALL FUNCTION 'DDIF_FIELDINFO_GET'
  EXPORTING
    TABNAME              = QUERY_TABLE
*   FIELDNAME            = ' '
*   LANGU                = SY-LANGU
*   LFIELDNAME           = ' '
*   ALL_TYPES            = ' '
*   GROUP_NAMES          = ' '
  IMPORTING
*   X030L_WA             =
    DDOBJTYPE            = TABLE_TYPE
*   DFIES_WA             =
*   LINES_DESCR          =
  TABLES
    DFIES_TAB            = TABLE_STRUCTURE
*   FIXED_VALUES         =
  EXCEPTIONS
    NOT_FOUND            = 1
    INTERNAL_ERROR       = 2
    OTHERS               = 3
          .
IF SY-SUBRC <> 0.
  RAISE TABLE_NOT_AVAILABLE.
ENDIF.
IF TABLE_TYPE = 'INTTAB'.
  RAISE TABLE_WITHOUT_DATA.
ENDIF.

* ---------------------------------------------------------------------
*  isolate first field of DATA as output field
*  (i.e. allow for changes to structure DATA!)
* ---------------------------------------------------------------------
DATA LINE_LENGTH TYPE I.
FIELD-SYMBOLS <D>.
ASSIGN COMPONENT 0 OF STRUCTURE DATA TO <D>.
DESCRIBE FIELD <D> LENGTH LINE_LENGTH in character mode.

* ---------------------------------------------------------------------
*  if FIELDS are not specified, read all available fields
* ---------------------------------------------------------------------
DATA NUMBER_OF_FIELDS TYPE I.
DESCRIBE TABLE FIELDS LINES NUMBER_OF_FIELDS.
IF NUMBER_OF_FIELDS = 0.
  LOOP AT TABLE_STRUCTURE.
    MOVE TABLE_STRUCTURE-FIELDNAME TO FIELDS-FIELDNAME.
    APPEND FIELDS.
  ENDLOOP.
ENDIF.
```

```
* ---------------------------------------------------------------------
*  for each field which has to be read, copy structure information
*  into tables FIELDS_INT (internal use) and FIELDS (output)
* ---------------------------------------------------------------------
DATA: BEGIN OF FIELDS_INT OCCURS 10,
        FIELDNAME  LIKE TABLE_STRUCTURE-FIELDNAME,
        TYPE       LIKE TABLE_STRUCTURE-INTTYPE,
        DECIMALS   LIKE TABLE_STRUCTURE-DECIMALS,
        LENGTH_SRC LIKE TABLE_STRUCTURE-INTLEN,
        LENGTH_DST LIKE TABLE_STRUCTURE-LENG,
        OFFSET_SRC LIKE TABLE_STRUCTURE-OFFSET,
        OFFSET_DST LIKE TABLE_STRUCTURE-OFFSET,
      END OF FIELDS_INT,
      LINE_CURSOR TYPE I.

LINE_CURSOR = 0.
*  for each field which has to be read ...
LOOP AT FIELDS.

  READ TABLE TABLE_STRUCTURE WITH KEY FIELDNAME = FIELDS-FIELDNAME.
  IF SY-SUBRC NE 0.
    RAISE FIELD_NOT_VALID.
  ENDIF.

* compute the place for field contents in DATA rows:
* if not first field in row, allow space for delimiter
  IF LINE_CURSOR <> 0.
    IF NO_DATA EQ SPACE AND DELIMITER NE SPACE.
      LINE_CURSOR = LINE_CURSOR + 1. "SARMA
      MOVE DELIMITER TO DATA+LINE_CURSOR .
    ENDIF.
    LINE_CURSOR = LINE_CURSOR + STRLEN( DELIMITER ).
  ENDIF.

* ... copy structure information into tables FIELDS_INT
* (which is used internally during SELECT) ...
  FIELDS_INT-FIELDNAME  = TABLE_STRUCTURE-FIELDNAME.
  FIELDS_INT-LENGTH_SRC = TABLE_STRUCTURE-INTLEN .
  FIELDS_INT-LENGTH_DST = TABLE_STRUCTURE-LENG  .
  FIELDS_INT-OFFSET_SRC = TABLE_STRUCTURE-OFFSET .
  FIELDS_INT-OFFSET_DST = LINE_CURSOR .
  FIELDS_INT-TYPE       = TABLE_STRUCTURE-INTTYPE.
  FIELDS_INT-DECIMALS   = TABLE_STRUCTURE-DECIMALS.
* compute the place for contents of next field in DATA rows
  LINE_CURSOR = LINE_CURSOR + TABLE_STRUCTURE-LENG.
  IF LINE_CURSOR > LINE_LENGTH AND NO_DATA EQ SPACE.
    RAISE DATA_BUFFER_EXCEEDED.
  ENDIF.
  APPEND FIELDS_INT.

* ... and into table FIELDS (which is output to the caller)
  FIELDS-FIELDTEXT = TABLE_STRUCTURE-FIELDTEXT.
  FIELDS-TYPE      = TABLE_STRUCTURE-INTTYPE.
  FIELDS-LENGTH    = FIELDS_INT-LENGTH_DST + 2 .
  FIELDS-OFFSET    = FIELDS_INT-OFFSET_DST + 2.
  MODIFY FIELDS.

ENDLOOP.
* end of loop at FIELDS

* ---------------------------------------------------------------------
*  read data from the database and copy relevant portions into DATA
* ---------------------------------------------------------------------
* output data only if NO_DATA equals space (otherwise the structure
* information in FIELDS is the only result of the module)
IF NO_DATA EQ SPACE.
```

```
DATA: BEGIN OF WORK, BUFFER(30000), END OF WORK.
FIELD-SYMBOLS: <WA> TYPE ANY, <COMP> TYPE ANY.
ASSIGN WORK TO <WA>> CASTING TYPE (QUERY_TABLE).
IF ROWCOUNT > 0.
  ROWCOUNT = ROWCOUNT + ROWSKIPS.
ENDIF.
SELECT * FROM (QUERY_TABLE) INTO <WA>WHERE (OPTIONS).

    IF SY-DBCNT GT ROWSKIPS.

*   copy all relevant fields into DATA (output) table
      LOOP AT FIELDS_INT.
        IF FIELDS_INT-TYPE = 'P'.
        ASSIGN COMPONENT FIELDS_INT-FIELDNAME
            OF STRUCTURE <WA> TO <COMP>
           TYPE FIELDS_INT-TYPE
            DECIMALS FIELDS_INT-DECIMALS.
        ELSE.
        ASSIGN COMPONENT FIELDS_INT-FIELDNAME
            OF STRUCTURE <WA> TO <COMP>
            TYPE     FIELDS_INT-TYPE.
            ENDIF.
            MOVE <COMP> TO
            <D>+FIELDS_INT-OFFSET_DST(FIELDS_INT-LENGTH_DST).
      ENDLOOP.
*   end of loop at FIELDS_INT
      APPEND DATA.
      IF ROWCOUNT > 0 AND SY-DBCNT GE ROWCOUNT. EXIT. ENDIF.

    ENDIF.

  ENDSELECT.

ENDIF.

ENDFUNCTION.
```

18. Go to SE37 tcode and select ZRFC_READ_TABLE. Click **Change**.
19. Click **Attributes** tab, select **Remote-Enabled Module** in the Processing Type pane.
20. Click **Save**.
21. Click **Activate**.
22. To configure the function you just created for Query interface for SAP Software in the J2C Bean wizard, specify the name of the custom function you created in step 1 on page 175, in the **Custom retrieve function name** field in the Configure Composite Properties window.

Results

The adapter retrieves an error-free data from the SAP tables during query interface.

SAP dependencies when using the WebSphere Adapter for SAP Software with the Advanced Event Processing (AEP) interface

The AEP transports require certain standard SAP function modules to import and work correctly. These standard function modules are required by the WebSphere BI Station Tool (SAPGUI transaction /CWLD/HOME_AEP), which is used to monitor and modify events for processing by the AEP interface

About this task

When using an SAP instance other than SAP ERP 4.7, ECC 5.0/ECC 6.0 (like SRM which differ significantly from ECC/ERP systems in terms of available standard SAP Function Modules), please ensure that you have the below listed SAP Standard Function Modules in your SAP instance(s) to run the adapter using the AEP (Advanced Event Processing) interface.

Procedure

1. AUTHORITY_CHECK_DATASET
2. BDC_CLOSE_GROUP
3. BDC_INSERT
4. BDC_OPEN_GROUP
5. BDC_RECORD_TRANSACTION
6. BP_FIND_JOBS_WITH_PROGRAM
7. BP_JOB_CREATE
8. BP_JOBLIST_PROCESSOR
9. DDIF_DTEL_GET
10. DDIF_FIELDINFO_GET
11. DDIF_NAMETAB_GET DYNP_VALUES_READ
12. DYNP_VALUES_UPDATE
13. EDI_DOCUMENT_STATUS_DISPLAY
14. ENQUEUE_READ
15. F4_USER
16. FUNCTION_DELETE
17. FUNCTION_EXISTS
18. FUNCTION_IMPORT_DOKU
19. HELP_VALUES_GET_WITH_TABLE
20. IDOCTYPE_READ
21. IMPORT_DYNPRO
22. NAME_OF_CURRENT_TRACE_FILE
23. NAMETAB_GET
24. NUMBER_GET_NEXT
25. POPUP_TO_CONFIRM
26. POPUP_TO_CONFIRM_LOSS_OF_DATA
27. POPUP_TO_CONFIRM_STEP
28. POPUP_TO_CONFIRM_WITH_MESSAGE
29. POPUP_TO_DECIDE
30. POPUP_TO_DECIDE_WITH_MESSAGE
31. POPUP_TO_DISPLAY_TEXT
32. POPUP_TO_GET_VALUE
33. POPUP_TO_INFORM
34. POPUP_WITH_3_BUTTONS_TO_CHOOSE
35. READ_TEXT
36. RS_CREATE_VARIANT
37. RS_FUNCTIONMODULE_INSERT

38. RS_TOOL_ACCESS
39. RS_TREE_CONSTRUCT
40. RS_TREE_LIST_DISPLAY
41. RS_TREE_SET_NODE
42. RS_VARIANT_ADD
43. RS_VARIANT_CATALOG
44. RS_VARIANT_CHECK_TSTC
45. RS_VARIANT_DELETE
46. SAPGUI_PROGRESS_INDICATOR
47. SEGMENTDEFINITION_READ
48. SQLT_GEN_TRACE_RECORDS (or) SQLT_GEN_TRACE_RECORDS_NEW
49. SQLT_GET_TRACE_RECORDS
50. SWO_OBJTYPE_EXIST
51. SWO_QUERY_API_METHODS
52. SWO_QUERY_BASEDATA
53. SWO_QUERY_KEYFIELDS
54. SWO_TYPE_INFO_GET
55. TABLE_CUSTOMIZING_MAINTENANCE
56. WS_FILENAME_GET
57. WS_QUERY

Results

Importing the transports on SAP systems will not fail if all of the above listed standard SAP Function Modules are present.

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 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.

Supported codepages for WebSphere Adapter for SAP Software

Since the adapter is based on Java, it automatically converts language characters to Unicode when sending data to the broker. It also automatically converts Unicode characters to the code page of the SAP system (if set to non-Unicode) and the SAP JCo support when sending data to the SAP system (if set to non-Unicode).

The adapter works with the SAP JCo API, which solely supports 8000 Shift-JIS. Refer to the following SAP notes for a list of supported code pages for data conversion.

- SAP note 794411
- SAP note 73606

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.

Avoiding stale connection problems in the SAP adapter

Connection related problems can be resolved in WebSphere Adapter for SAP Software using two new properties, **connectionRetryLimit** and **connectionRetryInterval** defined in the Managed connection factory of the SAP adapter.

Before you begin

The two properties are used to provide two features during outbound communication of the adapter and are optional.

About this task

In the service generation and deployment configuration window explained in the Setting deployment properties and generating the service topic for the appropriate interface,

Procedure

1. Click on the **Advanced -> Additional connection configuration**
2. Set the **Maximum number of retries in case of system connection failure** to the appropriate positive integer
   - if the property value is set to 0, then adapter does not perform any EIS connection validation and executes the outbound operation. If the EIS connection is invalid, the outbound operation fails. Though the subsequent requests get executed successfully provided the SAP system is functional, the current request fails.
   - if the property value is set to greater than 0, then during each request the adapter validates that the EIS connection is active/alive .

– If connection is valid then operation is completed. if connection is invalid, the adapter invalidates the current managed connection so that a new managed connection is created (new physical connection).
– If the connection is created successfully, the outbound operation is completed otherwise a `ResourceException` error is thrown.

3. Set the **Time interval between retries if connection fails (milliseconds)** field with the appropriate integer to indicate time in milliseconds in between retries. This property is enabled only when the **connectionRetryLimit** property has a value greater than 0.

Results

The two new features take care of connections which are timed out or become stale after the EIS restarts. Even though this option solves most of connection related problems, the adapter is not guaranteed to work 100% error free for connection problems.

Related reference:

"Managed connection factory properties" on page 219
Managed connection factory properties are used by the adapter at run time to create an outbound connection instance with the SAP server.

Resolving selector exception error

Selector exception errors can be resolved in WebSphere Adapter for SAP Software by identifying the IDoc which has not been configured. Using the J2C Bean wizard, configure the IDoc to resolve the error.

About this task

For Ale Inbound and BAPI inbound

Procedure

1. If you configure the adapter to work for a specified IDoc/ BAPI (eg., ALEREQ01.BAPI BAPI_CUSTOMER_GETLIST, etc.)
2. A different IDoc is sent from the SAP system (eg., ORDERS05), a function selector exception is logged which is displayed as follows:
   - For ALE inbound:

     Business object definition for 'SapOrders05' in namespace `'http://www.ibm.com/xmlns/prod/websphere/j2ca/sap/saporders05'` not found. The adapter has not been configured for the IDoc Type ORDERS05
   - For BAPI inbound:

     Business object definition for 'SapBapiCustomerGetlistWrapper' in namespace `'http://www.ibm.com/xmlns/prod/websphere/j2ca/sap/sapbapicustomergetlistwrapper'` not found. The adapter has not been configured for the BAPI BAPI_CUSTOMER_GETLIST

Results

To resolve the function selector exception, run the EMD again and select the particular IDoc. Refer to "Configuring the module for inbound processing" on page 129 to configure the adapter for the selected IDoc.

Related tasks:

"Configuring a module for ALE inbound processing" on page 136
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.

"Configuring a module for BAPI inbound processing" on page 130
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.

Resolving a service 'sapxxnn' unknown error

While deploying an inbound module to test environment, if you get an error message stating that the "service 'sapxxnn' unknown" (where xx is two letters, and nn is two numbers), you can resolve this by adding a missing entry into the services file.

This error is caused when the SAP gateway protocol 'sapxxnn' is not existing in the services file. In UNIX, it is located in **etc/services** and in Windows, it is located in **\WINDOWS\system32\drivers\etc\services**. To verify and resolve this error:

- Open the services file in a text editor (notepad or wordpad) and search for a valid 'sapxxnn' entry.
- If it is not in the services file, add the line 'sapxxnn' (minus the quotes) at the bottom of the file. There must be a carriage return at the end of this line if it is the last line in the file.
- You also need to add **port/protocol** in addition to the 'sapxxnn' entry. For example, **sapgw00 3600/tcp**

Resolving SAP JCo environment setup errors

When deploying a module (Inbound or Outbound) using any interface with the JCo3 jar, the module deployment may fail with the `NoClassDefFoundError` or a `ClassNotFoundException` in the run time environment. The error can be resolved by making sure the `CWYAP_SAPAdapterExt.jar`, which is a mandatory JAR, is available in the WebSphere Application Server run time class path.

When using JCo, the `CWYAP_SAPAdapterExt.jar` is a mandatory JAR and must be available in the WebSphere Application Server run time classpath. If the WebSphere Application Server run time cannot find the JAR, you will see a stacktrace in the trace/log as shown below:

```
Caused by: java.lang.ClassNotFoundException: com.ibm.j2ca.sap.ext.JCo3DestinationDataProvider
 at java.net.URLClassLoader.findClass(URLClassLoader.java:496)
 at com.ibm.ws.bootstrap.ExtClassLoader.findClass(ExtClassLoader.java:132)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:631)
 at com.ibm.ws.bootstrap.ExtClassLoader.loadClass(ExtClassLoader.java:87)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:597)
```

To resolve this error, perform the following steps:

1. Stop the WebSphere Application Server instance
2. Copy the `CWYAP_SAPAdapterExt.jar` JAR into the `$WPS_Root\lib\ext` folder

3. Restart WebSphere Application Server and deploy the module again.

Error during inbound processing

While processing ALE or BAPI inbound events that are deployed on WebSphere Application Server runtime, there is an error displayed that indicates that messages cannot be dumped into the log file. However, this event is successfully processed by the adapter.

This is a known issue and it does not have any functionality impact on the working of the adapter. This error can be ignored. For more details contact IBM support.

The error trace in the trace/log is as shown below:

```
[4/12/10 8:01:38:906 GMT+08:00] 0000002a SAPRAResource F com.ibm.j2ca.extension.utils.persistencesto
[4/12/10 8:01:38:906 GMT+08:00] 0000002a ManagerAdmin  W   TRAS0030W: Failed to dump ring buffer to s
 at com.ibm.ejs.ras.RasHelper.createFileOutputStream(RasHelper.java:1064)
 at com.ibm.ejs.ras.ManagerAdmin.dumpEmptyBuffer(ManagerAdmin.java:783)
 at com.ibm.ejs.ras.ManagerAdmin.dumpRingBuffer(ManagerAdmin.java:753)
 at com.ibm.ws.logging.WsLogger.dumpRingBuffer(WsLogger.java:1168)
 at com.ibm.ws.logging.WsLogger.deliverOrBuffer(WsLogger.java:294)
 at com.ibm.ws.logging.WsLogger.log(WsLogger.java:269)
 at com.ibm.j2ca.extension.logging.LogUtils.logDetails(LogUtils.java:1273)
 at com.ibm.j2ca.extension.logging.LogUtils.log(LogUtils.java:632)
 at com.ibm.j2ca.extension.logging.LogUtils.log(LogUtils.java:691)
 at com.ibm.j2ca.extension.utils.persistencestore.EventPersistenceMemoryImpl.logEpFatal(EventPersist
 at com.ibm.j2ca.extension.utils.persistencestore.EventPersistenceMemoryImpl.getEventStatus(EventPer
 at com.ibm.j2ca.extension.utils.persistencestore.EventPersistence.getEventStatus(EventPersistence.ja
 at com.ibm.j2ca.sap.inbound.eventrecovery.EventRecoveryManager.getEventStatus(EventRecoveryManager.
 at com.ibm.j2ca.sap.inbound.SAPTransactionalEventListener.checkTID(SAPTransactionalEventListener.jav
 at com.ibm.j2ca.sap.JCo3ServerTIDHandler.checkTID(JCo3ServerTIDHandler.java:26)
 at com.sap.conn.jco.rt.DefaultServerWorker.onCheckTID(DefaultServerWorker.java:209)
 at com.sap.conn.jco.rt.MiddlewareJavaRfc$JavaRfcServer.playbackTRfc(MiddlewareJavaRfc.java:2613)
 at com.sap.conn.jco.rt.MiddlewareJavaRfc$JavaRfcServer.handletRfcRequest(MiddlewareJavaRfc.java:254
 at com.sap.conn.jco.rt.MiddlewareJavaRfc$JavaRfcServer.listen(MiddlewareJavaRfc.java:2365)
 at com.sap.conn.jco.rt.DefaultServerWorker.dispatch(DefaultServerWorker.java:278)
 at com.sap.conn.jco.rt.DefaultServerWorker.loop(DefaultServerWorker.java:363)
 at com.sap.conn.jco.rt.DefaultServerWorker.run(DefaultServerWorker.java:239)
 at java.lang.Thread.run(Thread.java:737)
Caused by: java.security.PrivilegedActionException: java.io.FileNotFoundException: D:\IBM\WAS80\AppSe
at com.ibm.ws.security.util.AccessController.doPrivileged(AccessController.java:122)
 at com.ibm.ejs.ras.RasHelper.createFileOutputStream(RasHelper.java:1054)
 ... 22 more
Caused by: java.io.FileNotFoundException: D:\IBM\WAS80\AppServer\profiles\AppSrv02\logs\server1Fatal
 at java.io.FileOutputStream.<init>(FileOutputStream.java:190)
  at java.io.FileOutputStream.<init>(FileOutputStream.java:113)
  at com.ibm.ejs.ras.RasHelper$1.run(RasHelper.java:1057)
 at com.ibm.ejs.ras.RasHelper$1.run(RasHelper.java:1055)
 at com.ibm.ws.security.util.AccessController.doPrivileged(AccessController.java:118)
 ... 23 more
```

Frequently Asked Questions

This section provides you with answers to common questions about WebSphere Adapter for SAP Software.

- " How to use the Reset JCO Client?" on page 185
- "What are the mandatory fields while deploying an adapter at the node level?" on page 185

How to use the Reset JCO Client?

The reset property (Reset the JCO client after closing the connection handle) is used to obtain the latest changes of the ABAP definition (BAPI) dynamically. The changes can be any logical change in the ABAP code or a parameter change in the function module. This property is applicable only for BAPI Outbound modules.

Following are the advantages of using the Reset JCO client feature:

Get the latest changes that happened to the SAP system by discarding the cached contents in JCO.

Get the changes that happened to BAPI definition dynamically without restarting the adapter.

What are the mandatory fields while deploying an adapter at the node level?

When you deploy the adapter at the node level and manually create the activation specification and the managed connection factory properties ensure that you specify the mandatory fields as indicated below.

For ALE inbound, the following example indicates the mandatory properties that you need to specify in the activation specification properties. You can locate the following properties in the **.export** file.

```
<properties>
 <BONamespace>http://www.ibm.com/xmlns/prod/websphere/j2ca/sap</BONamespace>
 <applicationServerHost>9.184.167.115applicationServerHost>9.184.167.115>
 <assuredOnceDelivery>falseassuredOnceDelivery>false>
 <client>100client>100>
 <gatewayHost>9.184.167.115gatewayHost>9.184.167.115>
 <gatewayService>sapgw10gatewayService>sapgw10>
 <language>ENlanguage>EN>
 <password>saperp03password>saperp03>
 <rfcProgramID>TESTRFCSERVERrfcProgramID>TESTRFCSERVER>
 <systemNumber>10systemNumber>10>
 <userName>TESTuserName>TEST>
</properties>
```

For BAPI inbound, the following example indicates the mandatory properties that you need to specify in the activation specification properties. You can locate the following properties in the **.export** file.

```
<properties>
    <BONamespace>http://www.ibm.com/xmlns/prod/websphere/j2ca/sap</BONamespace>
    <applicationServerHost>9.184.167.115applicationServerHost>9.184.167.115>
    <assuredOnceDelivery>falseassuredOnceDelivery>false>
    <client>100client>100>
    <gatewayHost>9.184.167.115gatewayHost>9.184.167.115>
    <gatewayService>sapgw10gatewayService>sapgw10>
    <language>EN (English)language>EN (English)>
    <password>saperp02password>saperp02>
    <rfcProgramID>TESTRFCSERVERrfcProgramID>TESTRFCSERVER>
    <systemNumber>10systemNumber>10>
    <userName>TESTuserName>TEST>
</properties>
```

For BAPI outbound, the following example indicates the mandatory properties that you need to specify in the activation specification properties. You can locate the following properties in the **.import** file.

```
<resourceAdapter>
<connection type="com.ibm.j2ca.sap.SAPManagedConnectionFactory" interactionType="com.ibm.j2ca.sap.SAI
 <properties>
  <adapterID>001<,adapterID>
  <applicationServerHost>9.184.167.115applicationServerHost>9.184.167.115>
  <language>ENlanguage>EN>
  <password>saperp02password>saperp02>
  <systemNumber>10M/systemNumber>
  <userName>TESTuserName>TEST>
 </properties>
</connection>
```

Support

This section provides information about how to troubleshoot a problem with your IBM® software, including instructions for searching knowledge bases, downloading fixes, and obtaining support.

Searching knowledge bases (Web search)

You can often find solutions to problems by searching IBM knowledge bases. You can optimize your results by using available resources, support tools, and search methods.

About this task

You can find useful information by searching the information center for Product X. However, sometimes you need to look beyond the information center to answer your questions or resolve problems.

To search knowledge bases for information that you need, use one or more of the following approaches:

- Search for content by using the IBM® Support Assistant (ISA).

  ISA is a no-charge software serviceability workbench that helps you answer questions and resolve problems with IBM software products. You can find instructions for downloading and installing ISA on the ISA website.
- Find the content that you need by using the IBM Support Portal.

  The IBM Support Portal is a unified, centralized view of all technical support tools and information for all IBM systems, software, and services. The IBM Support Portal lets you access the IBM electronic support portfolio from one place. You can tailor the pages to focus on the information and resources that you need for problem prevention and faster problem resolution. Familiarize yourself with the IBM Support Portal by viewing the demo videos (https://www.ibm.com/blogs/SPNA/entry/the_ibm_support_portal_videos) about this tool. These videos introduce you to the IBM Support Portal, explore troubleshooting and other resources, and demonstrate how you can tailor the page by moving, adding, and deleting portlets.
- Search for content by using the IBM masthead search. You can use the IBM masthead search by typing your search string into the Search field at the top of any ibm.com® page.
- Search for content by using any external search engine, such as Google, Yahoo, or Bing. If you use an external search engine, your results are more likely to include information that is outside the ibm.com domain. However, sometimes you can find useful problem-solving information about IBM products in newsgroups, forums, and blogs that are not on ibm.com.

Tip: Include "IBM" and the name of the product in your search if you are looking for information about an IBM product.

Getting Fixes

A product fix might be available to resolve your problem.

About this task

To get product fixes, perform the following steps.

Procedure

1. Determine which fix you need. Check the list of IBM WebSphere Adapter for SAP Software recommended fixes to confirm that your software is at the latest maintenance level. Check the list of problems fixed in the IBM WebSphere Adapter for SAP Software fix readme documentation that is available for each listed fix pack to see if IBM has already published an individual fix to resolve your problem. To determine what fixes are available using IBM Support Assistant, run a query on fix from the search page.

   Individual fixes are published as often as necessary to resolve defects in WebSphere Application Server IBM WebSphere Adapter for SAP Software. In addition, two kinds of cumulative collections of fixes, called fix packs and refresh packs, are published periodically for IBM WebSphere Adapter for SAP Software, in order to bring users up to the latest maintenance level. You should install these update packages as early as possible in order to prevent problems.

   Note: A list of recommended, generally available (GA) fixes for the WebSphere Java™ Connector Architecture (JCA) and WebSphere Business Integration adapters are available here. If a Fix Pack is not available for an adapter, it implies that the GA version is the recommended version and details about that version of the adapter can be found in the Release notes.
2. Download the fix. Open the download document and follow the link in the Download package section. When downloading the file, ensure the name of the maintenance file is not changed. This includes both intentional changes and inadvertent changes caused by certain web browsers or download utilities.
3. Apply the fix. Follow the instructions in the Installation Instructions section of the download document.
4. Optional: To receive weekly notification of fixes and updates, subscribe to My Support e-mail updates.

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 website

The WebSphere Adapters software support website at http://www-947.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_Adapters_Family provides links to many resources to help you learn about, use, and troubleshoot WebSphere Adapters, including:

- 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 must 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 WebSphere 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 Adapter for SAP Software, see http://www-01.ibm.com/support/docview.wss?uid=swg27024045.

For a list of technotes for all adapters, see http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8 &dc=DB520+D800+D900+DA900+DA800+DB560&dtm.

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 22. 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 22. 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. **Note:** This is applicable if the **Generate BAPIs within Wrappers** check box is selected. If you do not select the **Generate BAPIs within Wrappers** check box, top-level business objects are automatically generated for each BAPI selected. The adapter internally assigns the **Execute** operation for each top-level business object generated.

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 23. 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 24. 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 25. 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`.

Table 25. Business object-level metadata elements: ALE business object (continued)

Metadata element	Description
Type	The business object type. Possible values are `IDOC` or `UNPARSEDIDOC`.
Operation	Each *outbound* operation contains the following parameters: **Name** Name of the operation: For outbound processing, it 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 tables describe the business-object metadata elements of an ALE pass-through IDoc interface business object.

Table 26. 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`.
Delimiter	Use a delimiter to split an IDoc's control record (fixed length) or IDoc segments (if less than specified length). The possible values are any strings without escape characters, `\\n` or `\\r\\n`. Input a delimiter in the wizard at the following location "Selecting business objects and services for ALE pass-through IDoc outbound processing" on page 110

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 27. 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.

Table 27. Property-level metadata elements: ALE business object (continued)

Metadata element	Description
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 28. 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 29. 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.

Table 29. Business object-level metadata elements: Query interface for SAP Software table business object (continued)

Metadata element	Description
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 30. 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 31. Business object-level metadata elements: Advanced event processing

Metadata element	Description
Type	The business object type. The business object type will always be `AEP`.
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 32. 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.

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 33. 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 34. Supported operations: BAPI business objects

Operation	Definition
Create	The top-level business object and all contained children are created.

Table 34. 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.
Execute	The top-level business object and any contained children is executed. **Note:** This operation is available only if the **Generate BAPIs within Wrappers** check box is not selected. If the Configure Wrapper Business Object for Selected BAPI check box is selected, then other operations, such as Create, Update, Delete, and Retrieve are available.

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 35. 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 36. 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 37. 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 38. 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.

Table 38. Supported operations: Query interface for SAP Software business objects (continued)

Operation	Description
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 39. 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 40. 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 41. 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`

When you select the option in the Discovery Configuration window, the adapter generates all the Business Object names same as that of the SAP with original casing. This enables the elements (at child and grand-child levels) which are complex type to appear in the SAP original casing.

If the module contains structures that have the same name, for example, the `RETURN` structure, the adapter handles Business Object name duplication depending on the value specified for the **Enforce the same naming convention for business objects** EMD property. The behaviour differs based on the selection in the following manner:

1. If the **Enforce the same naming convention for business objects** checkbox is checked, the adapter generates all the Business Object name without appending any hashcode. The hashcode is appended to the namespace of each Business Object instead of the name.

   Example: If the repeating structure is RETURN, the adapter generates the corresponding Business Object name SapReturn. If there are three occurrences of the RETURN structure in the same module, then all three Business Objects will have the name SapReturn.
2. If the **Enforce the same naming convention for business objects** checkbox is unchecked, the adapter appends the hashcode to the subsequent Business Object that has the same name. This is done to avoid duplication.

   Example: If the repeating structure is RETURN, the adapter generates the corresponding Business Object name SapReturn. If there are three occurrences of the RETURN structure in the same module,
   - The first occurrence of the SapReturn Business Object has the name SapReturn.
   - The second occurrence of the SapReturn Business Object has the name SapReturn203510914 where 203510914 is the hashcode of the String SapReturn.
   - The third occurrence of the SapReturn Business Object has name SapReturn619647890 where 619647890 is the hashcode of the String SapReturn203510914.

BAPI work unit

The following table describes the convention applied by the J2C Bean wizard when naming a BAPI work unit business object.

Table 42. 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 the module contains structures that have the same name, for example, the RETURN structure, the adapter handles Business Object name duplication depending on the value specified for the **Enforce the same naming convention for business objects** EMD property. The behaviour differs based on the selection in the following manner:

1. If the **Enforce the same naming convention for business objects** checkbox is checked, the adapter generates all the Business Object name without appending any hashcode. The hashcode is appended to the namespace of each Business Object instead of the name.

Example: If the repeating structure is RETURN, the adapter generates the corresponding Business Object name SapReturn. If there are three occurrences of the RETURN structure in the same module, then all three Business Objects will have the name SapReturn.

2. If the **Enforce the same naming convention for business objects** checkbox is unchecked, the adapter appends the hashcode to the subsequent Business Object that has the same name. This is done to avoid duplication.

   Example: If the repeating structure is RETURN, the adapter generates the corresponding Business Object name SapReturn. If there are three occurrences of the RETURN structure in the same module,

   - The first occurrence of the SapReturn Business Object has the name SapReturn.
   - The second occurrence of the SapReturn Business Object has the name SapReturn203510914 where 203510914 is the hashcode of the String SapReturn.
   - The third occurrence of the SapReturn Business Object has name SapReturn619647890 where 619647890 is the hashcode of the String SapReturn203510914.

BAPI result sets

The following table describes the convention applied by the J2C Bean wizard when naming a BAPI result sets business object.

Table 43. 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 the module contains structures that have the same name, for example, the RETURN structure, the adapter handles Business Object name duplication depending on the value specified for the **Enforce the same naming convention for business objects** EMD property. The behaviour differs based on the selection in the following manner:

1. If the **Enforce the same naming convention for business objects** checkbox is checked, the adapter generates all the Business Object name without appending any hashcode. The hashcode is appended to the namespace of each Business Object instead of the name.

   Example: If the repeating structure is RETURN, the adapter generates the corresponding Business Object name SapReturn. If there are three occurrences of the RETURN structure in the same module, then all three Business Objects will have the name SapReturn.

2. If the **Enforce the same naming convention for business objects** checkbox is unchecked, the adapter appends the hashcode to the subsequent Business Object that has the same name. This is done to avoid duplication.

   Example: If the repeating structure is `RETURN`, the adapter generates the corresponding Business Object name `SapReturn`. If there are three occurrences of the `RETURN` structure in the same module,

   - The first occurrence of the `SapReturn` Business Object has the name `SapReturn`.
   - The second occurrence of the `SapReturn` Business Object has the name `SapReturn203510914` where 203510914 is the hashcode of the String `SapReturn`.
   - The third occurrence of the `SapReturn` Business Object has name `SapReturn619647890` where 619647890 is the hashcode of the String `SapReturn203510914`.

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 44. 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 44. 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`

Table 44. Naming conventions for ALE business objects (continued)

Element	Naming convention
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 45. 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 46. 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

IBM 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 processes the field using its assigned default value, and that default value is 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 notes 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 states 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: • 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), code page 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 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. 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 205.

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 47. External service connection properties Adapter for SAP Software

Property name	Description
"Bidi direction " on page 208	The orientation component of the bidi format specification.
"Bidi ordering schema" on page 209	The ordering scheme of the bidi format specification.
"Bidi numeric shaping" on page 209	The numeric shaping component of the bid format specification.
"Bidi shaping" on page 209	The shaping component of the bidi format specification.
"Bidi symmetric swapping" on page 210	The symmetric swapping component of the bid format specification.
"Client" on page 210	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 210	Indicates the numeric identifier of the code page.
"Folder for RFC trace files" on page 211	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Host name" on page 211	Specifies the IP address or the name of the application server host that the adapter logs on to.
"Language code" on page 212	Specifies the language in which the adapter logs on.
"Log file output location property" on page 212	Specifies the location of the log file for external service.
"Logging level property" on page 212	Specifies the type error for which logging will occur during external service.

Table 47. External service connection properties Adapter for SAP Software (continued)

Property name	Description
"Password" on page 213	The password of the user account of the adapter on the SAP application server.
"RFC trace level" on page 214	Specifies the global trace level.
"RFC trace on" on page 214	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP interface name" on page 215	Indicates the SAP interface to be used.
"System number" on page 216	The system number of the SAP application server.
"User name" on page 216	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 48. 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 an 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 an RTL document the character will become RTL).
Default	`LTR`
Property type	String

Table 48. 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 49. 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 50. 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 51. Bidi shaping details

Required	No

Possible values	`Nominal`
	`Shaped`
	`Initial`
	`Middle`
	`Final`
	`Isolated`

Default	`Nominal`

Table 51. 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 52. 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 53. 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 54. 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 54. 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 55. 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. This field cannot be edited if you are modifying existing artifacts
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 56. 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 57. 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 J2C bean.

Table 58. 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 59. Logging level details

Required	No

Table 59. 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, such as 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 60. Password details

Required	Yes
Default	No default value
Property type	String

Table 60. 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 61. RFC trace level details

Required	No
Possible values	`0` - No error `1` - Errors and warnings `2` - Execution path, errors and warnings `3` - Full Execution path, errors and warnings `4` - Execution path, info messages, errors and warnings `6` - Full execution path, info messages, errors and warnings `7` - Debug messages, full execution path, info messages, errors and warnings `8` - Verbose debug messages, full execution path, info messages, errors and warnings
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 62. RFC trace on details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean

Table 62. 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 63. 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. This field cannot be edited if you are modifying existing artifacts
Globalized	No
Bidi supported	No

System number

This property is the system number of the SAP application server.

Table 64. 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 65. 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 205.

Table 66. 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 `X` strings instead of user data in the log and trace files.
(Not available)	"Enable high availability support (enableHASupport)" on page 219	Do not change this property.
(Not available)	LogFileSize	Deprecated
(Not available)	LogFilename	Deprecated
(Not available)	LogNumberOfFiles	Deprecated
(Not available)	TraceFileSize	Deprecated
(Not available)	TraceFileName	Deprecated
(Not available)	TraceNumberOfFiles	Deprecated

Adapter ID (AdapterID)

This property identifies a specific deployment or instance of the adapter.

Table 67. Adapter ID details

Required	Yes
Default	`001`
Property type	String

Table 67. Adapter ID details (continued)

Usage	This property identifies the adapter instance in the 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, ensure that the first eight characters of the adapter ID property are unique for each instance so that you can correlate the 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 the log and trace information to a particular instance of an adapter. For example, when 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. If you set the adapter ID properties of two instances to `Instance01` and `Instance02`, you will not be able to 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, the value can be set both at the resource adapter level and the managed connection factory level. After you use 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)

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 68. 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, the value can be set both at the resource adapter level and the managed connection factory level. After you use 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 6.2.x applies to the interaction specification property group in version 7.5.0.1.

- IgnoreBAPIReturn

You set the managed connection factory properties using the J2C Bean wizard and can change them using the J2C bean that is generated in 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 205.

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 69. Managed connection factory 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.
"Client" on page 221	Client	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 222	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 `X` strings instead of user data in the log and trace files.
"Maximum Number of retries in case of system connection failure" on page 223	connectionRetryLimit	The adapter will try connecting to the Enterprise Information System (EIS) for a specified number of tries. Select only if you want to reduce the number of connection exceptions in the outbound operation. If selected, adapter will validate the connection for each outbound request.
"Enable Secure Network Connection" on page 228	SncMode	Indicates whether secure network connection mode is used.
"Folder for RFC trace files" on page 223	RfcTracePath	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Gateway host" on page 224	GatewayHost	The host name of the SAP gateway.
"Gateway service" on page 224	GatewayService	The identifier of the gateway on the gateway host that carries out the RFC services.

Table 69. Managed connection factory properties for Adapter for SAP Software (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
"Host name" on page 225	ApplicationServerHost	Specifies the IP address or the name of the application server host that the adapter logs on to.
"Language code" on page 225	Language code	Specifies the Language code in which the adapter logs on to SAP.
"Load Balancing" on page 223	loadBalancing	Specifies if your SAP configuration uses load balancing
"Message server host" on page 225	MessageServerHost	Specifies the name of the host on which the message server is running.
"Partner character set" on page 226	PartnerCharset	Specifies PartnerCharset encoding.
"Password" on page 226	Password	The password of the user account of the adapter on the SAP application server.
"Reset JCo Client after closing Connection Handle" on page 226	resetClient	This property optionally invokes the reset method on the JCo client to make sure changes on the SAP EIS are reflected in the client during an outbound transaction
"RFC trace level" on page 227	RfcTraceLevel	Specifies the global trace level.
"RFC trace on" on page 227	RfcTraceOn	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP system ID" on page 228	SAPSystemID	Specifies the system ID of the SAP system for which logon load balancing is allowed.
"Secure Network Connection library path" on page 229	SncLib	Specifies the path to the library that provides the secure network connection service.
"Secure Network Connection name" on page 229	SncMyname	Specifies the name of the secure network connection.
"Secure Network Connection partner" on page 229	SncPartnername	Specifies the name of the secure network connection partner.
"Secure Network Connection security level" on page 230	SncQop	Specifies the level of security for the secure network connection.
"System number" on page 230	SystemNumber	The system number of the SAP application server.
"Time between retries in case of system connection failure (milliseconds)" on page 230	connectionRetryInterval	Specifies the time interval between attempts to restart the event listeners.
"User name" on page 231	userName	The user account for the adapter on the SAP server.
"Wait until commit call to SAP database is completed and returned" on page 231	WaitOnCommit	Enables adapter to wait until all time critical updates to SAP database are completed before calling commit on it.
"X509 certificate" on page 231	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 70. Adapter ID details

Required	Yes
Default	`001`
Property type	String
Usage	This property identifies the adapter instance in the 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, ensure that the first eight characters of the adapter ID property are unique for each instance so that you can correlate the 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 the log and trace information to a particular instance of an adapter. For example, when 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. If you set the adapter ID properties of two instances to `Instance01` and `Instance02`, you will not be able to 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, the value can be set both at the resource adapter level and the managed connection factory level. After you use 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

Client

This property is the client number of the SAP system to which the adapter connects.

Table 71. 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 72. 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

Disguise user data as "XXX" in log and trace files (HideConfidentialTrace)

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 73. 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, the value can be set both at the resource adapter level and the managed connection factory level. After you use 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

Load Balancing

This property specifies if your SAP configuration uses load balancing

Table 74. Load balancing details

Required	Yes
Possible values	`TrueFalse`
Default	False
Property type	Boolean
Usage	This value should be set to true if the SAP configuration uses load balancing. If set to true, Message server host, Logon group and SAP System ID need to be specified.
Globalized	No
Bidi supported	No

Maximum Number of retries in case of system connection failure

This property specifies the number of times the adapter attempts to create a connection to the Enterprise Information System (EIS). The adapter will try connecting to the EIS for the specified number of times. Select only if you want to reduce the number of connection exceptions in the outbound operation. If selected, the adapter will validate the connection for each outbound request.

Table 75. Reset Client details

Required	No
Possible values	Integers
Default	`0`
Property type	Integer
Usage	Only positive values are valid. When the adapter encounters an error related to the outbound connection, it retries to establish a physical connection (when physical connection is not established) for the number of times specified for this property with a time delay specified in the property "Time between retries in case of system connection failure (milliseconds)" on page 230. If the value is `0`, the adapter does not perform any EIS connection validation and executes the outbound operation. if the value is `> 0`, then during each request the adapter validates if the EIS connection is active. • If the connection is valid the operation is completed. • if connection is invalid, the adapter invalidates the current managed connection and a new managed connection is created (new physical connection)
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 76. 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. This field cannot be edited if you are modifying existing artifacts
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 77. 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 78. 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 79. 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 80. 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

Message server host

This property specifies the name of the host on which the message server is running.

Table 81. Message server host details

Required	Yes (if load balancing is used)
Default	No default value
Property type	String

Table 81. Message server host details (continued)

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 82. 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 83. 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

Reset JCo Client after closing Connection Handle

This property optionally invokes the reset method on the JCo client to make sure changes on the SAP EIS are reflected in the client during an outbound transaction

Table 84. Reset Client details

Required	No

Table 84. Reset Client details (continued)

Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	When the property is set to `True`, the adapter invokes the reset method on the JCo client to make sure changes on the SAP EIS are reflected in the client during an outbound transaction. When this property is set to `False` the adapter does not invoke the reset method on the JCo client, therefore any changes on the SAP EIS are not reflected in the client during an outbound transaction. This property is supported on the ALE interfaces - ALE and ALE Passthrough, and the BAPI interfaces - simple BAPI, BAPI work unit and BAPI ResultSet.
Globalized	NA
Bidi supported	No

RFC trace level

This property specifies the global trace level.

Table 85. RFC trace level details

Required	No
Possible values	`0` - No error `1` - Errors and warnings `2` - Execution path, errors and warnings `3` - Full Execution path, errors and warnings `4` - Execution path, info messages, errors and warnings `6` - Full execution path, info messages, errors and warnings `7` - Debug messages, full execution path, info messages, errors and warnings `8` - Verbose debug messages, full execution path, info messages, errors and warnings
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 86. RFC trace on details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean

Table 86. 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 87. 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

Enable Secure Network Connection

This property indicates whether secure network connection mode is enabled.

Table 88. 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: • SncLib • SncMyname • SncPartnername • SncQop

Table 88. Enable Secure Network Connection details (continued)

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 89. 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 90. 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 91. 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 92. 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 93. 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 connect to the Enterprise Information System (EIS).

Table 94. Time between retries in case of system connection failure details

Required	No
Possible Values	Positive Integers
Default	60000
Unit of measure	Milliseconds
Property type	Integer
Usage	When the adapter encounters an error related to the outbound connection, this property specifies the time interval that the adapter waits in between attempts to reestablish an outbound connection. It is disabled by default and is only enabled when the value of "Maximum Number of retries in case of system connection failure" on page 223 is greater than 0.

Table 94. Time between retries in case of system connection failure details (continued)

Globalized	No
Bidi supported	No

User name

This property is the user account for the adapter on the SAP server.

Table 95. 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

Wait until commit call to SAP database is completed and returned

For BAPI outbound processing, this property specifies whether the commit call from the adapter should wait until all the time-critical (V1) updates to the SAP database have been completed. When this property is set to true, the commit to call to SAP waits until all updates are completed and returned. If it is set to false, the commit call returns immediately from the SAP database.

This property is only active when you use the `CWYAP_SAPAdapter_Tx.rar` file.

Table 96. WaitOnCommit property details

Required	No
Default	`False`
Property type	Boolean
Globalized	No
Bidi supported	No

X509 certificate

This property specifies the X509 certificate to be used as the logon ticket.

Table 97. X509 certificate details

Required	No.
Default	No default value
Property type	String

Table 97. X509 certificate details (continued)

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 98 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 205.

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 J2C bean generated in theRational Application Developer for WebSphere Software administrative console .

Table 98. Interaction specification properties for Adapter for SAP Software

Property name		
In the wizard		**Description**
"Custom retrieve function name"		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		Populates the function name for the specific SAP interface.
Ignore errors in BAPI return		Indicates if errors in BAPI returns will be ignored.
"Maximum number of hits for the discovery" on page 235		Maximum number of result sets to return during a RetrieveAll operation.
"Select the queue name" on page 235		The name of a customer-defined queue on the SAP server.
"Wait until commit call to SAP database is completed and returned" on page 236		Enables adapter to wait until all time critical updates to SAP database are completed before calling commit on it.

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 99. Custom retrieve function name details

Required	No
Default	No default value
Property type	String

Table 99. Custom retrieve function name details (continued)

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 100. Function name details

Required	Yes
Possible values	`True` `False`
Default	Null
Property type	String

Table 100. Function name details (continued)

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 101. Ignore errors in BAPI return details

Required	No

Table 101. Ignore errors in BAPI return details (continued)

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 102. 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 103. 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
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

Table 103. Select the queue name details (continued)

Bidi supported	No

Wait until commit call to SAP database is completed and returned

For BAPI outbound processing, this property specifies whether a commit call from the adapter should wait until all the time-critical (V1) updates to the SAP database have been completed. When this property is set to true, the commit to call to SAP waits until all updates are completed and returned. If it is set to false, the commit call returns immediately from the SAP database.

This property is only active when you use the `CWYAP_SAPAdapter.rar` file.

Table 104. WaitOnCommit property details

Required	No
Default	`False`
Property type	Boolean
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 processes the field using its assigned default value, and that default value is 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 notes 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 states 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: • 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), code page 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 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. 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 205.

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 105. External service connection properties Adapter for SAP Software

Property name	Description
"Bidi direction " on page 239	The orientation component of the bidi format specification.
"Bidi ordering schema" on page 240	The ordering scheme of the bidi format specification.
"Bidi numeric shaping" on page 240	The numeric shaping component of the bid format specification.
"Bidi shaping" on page 240	The shaping component of the bidi format specification.
"Bidi symmetric swapping" on page 241	The symmetric swapping component of the bid format specification.
"Client" on page 241	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 241	Indicates the numeric identifier of the code page.
"Folder for RFC trace files" on page 242	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Host name" on page 242	Specifies the IP address or the name of the application server host that the adapter logs on to.
"Language code" on page 243	Specifies the language in which the adapter logs on.
"Log file output location property" on page 243	Specifies the location of the log file for external service.
"Logging level property" on page 243	Specifies the type error for which logging will occur during external service.

Table 105. External service connection properties Adapter for SAP Software (continued)

Property name	Description
"Password" on page 244	The password of the user account of the adapter on the SAP application server.
"RFC trace level" on page 245	Specifies the global trace level.
"RFC trace on" on page 245	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP interface name" on page 246	Indicates the SAP interface to be used.
"System number" on page 247	The system number of the SAP application server.
"User name" on page 247	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 106. 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 an 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 an RTL document the character will become RTL).
Default	`LTR`
Property type	String

Table 106. 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 107. 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 108. 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 109. Bidi shaping details

Required	No

Possible values	`Nominal`
	`Shaped`
	`Initial`
	`Middle`
	`Final`
	`Isolated`

Default	`Nominal`

Table 109. 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 110. 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 111. 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 112. 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 112. 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 113. 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. This field cannot be edited if you are modifying existing artifacts
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 114. 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 115. 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 J2C bean.

Table 116. 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 117. Logging level details

Required	No

Table 117. 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, such as 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 118. Password details

Required	Yes
Default	No default value
Property type	String

Table 118. 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 119. RFC trace level details

Required	No
Possible values	`0` - No error `1` - Errors and warnings `2` - Execution path, errors and warnings `3` - Full Execution path, errors and warnings `4` - Execution path, info messages, errors and warnings `6` - Full execution path, info messages, errors and warnings `7` - Debug messages, full execution path, info messages, errors and warnings `8` - Verbose debug messages, full execution path, info messages, errors and warnings
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 120. RFC trace on details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean

Table 120. 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 121. 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. This field cannot be edited if you are modifying existing artifacts
Globalized	No
Bidi supported	No

System number

This property is the system number of the SAP application server.

Table 122. 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 123. 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 205.

Table 124. Resource adapter properties for Adapter for SAP Software

Property name		
In the wizard	**In the administrative console**	**Description**
“Adapter ID (AdapterID)”	AdapterID	Identifies the adapter instance for PMI events and for logging and tracing.
“Disguise user data as "XXX" in log and trace files (HideConfidentialTrace) ” on page 249	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing `X` strings instead of user data in the log and trace files.
(Not available)	“Enable high availability support (enableHASupport)” on page 250	Specifies if only one instance of the adapter is active, or more than one adapter instance is active at a given time.
(Not available)	LogFileSize	Deprecated
(Not available)	LogFilename	Deprecated
(Not available)	LogNumberOfFiles	Deprecated
(Not available)	TraceFileSize	Deprecated
(Not available)	TraceFileName	Deprecated
(Not available)	TraceNumberOfFiles	Deprecated

Adapter ID (AdapterID)

This property identifies a specific deployment or instance of the adapter.

Table 125. Adapter ID details

Required	Yes
Default	`001`
Property type	String

Table 125. Adapter ID details (continued)

Usage	This property identifies the adapter instance in the 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, ensure that the first eight characters of the adapter ID property are unique for each instance so that you can correlate the 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 the log and trace information to a particular instance of an adapter. For example, when 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. If you set the adapter ID properties of two instances to `Instance01` and `Instance02`, you will not be able to 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, the value can be set both at the resource adapter level and the managed connection factory level. After you use 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)

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 126. 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, the value can be set both at the resource adapter level and the managed connection factory level. After you use 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)

This property is `true` by default and appears in the administrative console. When it is true, all Inbound applications will be started on all the cluster members but only one application will actively receive the events. The rest will be on stand by for fail over. When the property is set to `false` all the applications deployed in a high availability environment will actively poll.

Note: For HA Active-Active configuration, this property must be set to `false` in the administrative console

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 JNDI, in the WebSphere Application Server administrative console.

Table 127 lists and describes the activation specification properties that apply to both synchronous RFC and asynchronous transactional RFC. Table 128 on page 252 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 205.

Table 127. Activation specification properties for BAPI inbound processing

Property name		
In the wizard	**In the administrative console**	**Description**
"Client" on page 253	Client	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 254	Codepage	Indicates the numeric identifier of the code page.
"Enable Secure Network Connection" on page 255	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 256	RfcTracePath	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Gateway host" on page 257	GatewayHost	The host name of the SAP gateway.
"Gateway service" on page 257	GatewayService	The identifier of the gateway on the gateway host that carries out the RFC services.

Table 127. Activation specification properties for BAPI inbound processing (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
"Host name" on page 258	ApplicationServerHost	Specifies the IP address or the name of the application server host that the adapter logs on to.
"Language code" on page 258	Language code	Specifies the Language code in which the adapter logs on to SAP.
"Logon group name" on page 258	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.
"Load Balancing" on page 259	loadBalancing	Specifies if your SAP configuration uses load balancing
"Maximum number of retries in case of system connection failure" on page 259	connectionRetryLimit	The adapter will try connecting to the Enterprise Information System (EIS) for a specified number of tries. Select only if you want to reduce the number of connection exceptions in the outbound operation. If selected, adapter will validate the connection for each outbound request.
"Message server host" on page 260	MessageServerHost	Specifies the name of the host on which the message server is running.
"Number of listeners" on page 260	NumberOfListeners	Specifies the number of event listeners that are to be started.
"Partner character set" on page 260	PartnerCharset	Specifies PartnerCharset encoding.
"Password" on page 261	Password	The password of the user account of the adapter on the SAP application server.
"Retry EIS connection on startup" on page 261	RetryConnectionOn Startup	Controls whether the adapter retries the connection to the EIS if it cannot connect at startup
"RFC program ID" on page 262	RfcProgramID	The remote function call identifier under which the adapter registers in the SAP gateway.
"RFC trace level" on page 262	RfcTraceLevel	Specifies the global trace level.
"RFC trace on" on page 263	RfcTraceOn	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP system ID" on page 263	SAPSystemID	Specifies the system ID of the SAP system for which logon load balancing is allowed.
"Secure Network Connection library path" on page 263	SncLib	Specifies the path to the library that provides the secure network connection service.
"Secure Network Connection name" on page 264	SncMyname	Specifies the name of the secure network connection.
"Secure Network Connection partner" on page 264	SncPartnername	Specifies the name of the secure network connection partner.
"Secure Network Connection security level" on page 264	SncQop	Specifies the level of security for the secure network connection.
"System number" on page 265	SystemNumber	The system number of the SAP application server.

Table 127. Activation specification properties for BAPI inbound processing (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
"Time between retries in case of system connection failure (milliseconds)" on page 265	connectionRetryInterval	Specifies the time interval between attempts to restart the event listeners.
"User name" on page 266	userName	The user account for the adapter on the SAP server.
"X509 certificate" on page 266	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 128. Additional activation specification properties for assured-once delivery

Property name		
In the wizard	**In the administrative console**	**Description**
"Assured once-only delivery "	AssuredOnceDelivery	Specifies whether to provide assured-once delivery for inbound events.
"Auto create event table" on page 253	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 255	EP_SchemaName	The schema used for automatically creating the event recovery table.
"Event recovery data source (JNDI) name" on page 255	EP_DataSource_JNDIName	The JNDI name of the data source configured for event recovery.
"Event recovery table name" on page 255	EP_TableName	The name of the event recovery table.
"Password used to connect to event data source" on page 261	EP_Password	The user password for connecting to the database.
"User name used to connect to event data source" on page 266	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 129. Assured once-only delivery details

Required	No
Default	`True`
Property type	Boolean

Table 129. 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

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 130. 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.
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 131. Client details

Required	Yes
Possible values	You can enter a range of values from `000` to `999`.
Default	`100`
Property type	Integer

Table 131. Client details (continued)

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 132. 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 133. 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 Secure Network Connection

This property indicates whether secure network connection mode is enabled.

Table 134. 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: • SncLib • SncMyname • SncPartnername • SncQop
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 135. 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 administrative console. 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 136. 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 137. 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 138. Folder for RFC trace files details

Required	No
Default	No default value

Table 138. 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. This field cannot be edited if you are modifying existing artifacts
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 139. 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 140. 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 141. 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 142. 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 143. 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 143. 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

Load Balancing

This property specifies if your SAP configuration uses load balancing

Table 144. Load balancing details

Required	Yes
Possible values	`TrueFalse`
Default	False
Property type	Boolean
Usage	This value should be set to true if the SAP configuration uses load balancing. If set to true, Message server host, Logon group and SAP System ID need to be specified.
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. If the "Retry EIS connection on startup" on page 261 property is set to true it also indicates the maximum number of times the adapter will retry the connection to the EIS if it cannot connect at startup.

Table 145. Maximum number of retries in case of system failure details

Required	No
Possible values	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. Negative values indicate that the adapter will not attempt to establish the inbound connection
Globalized	No
Bidi supported	No

Message server host

This property specifies the name of the host on which the message server is running.

Table 146. 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 147. 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. **Note:** The adapter will not start if the number of listeners is 0
Globalized	No
Bidi supported	No

Partner character set

This property specifies the partner character set encoding.

Table 148. 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 149. 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

This property controls whether the adapter retries the connection to the EIS if it cannot connect at startup. This property is used in conjunction with "Maximum number of retries in case of system connection failure" on page 259 and "Time between retries in case of system connection failure (milliseconds)" on page 265.

Table 150. Retry EIS connection on startup

Required	No

Possible Values	`True`
	
	`False`

Default	`False`
Property type	Boolean
Usage	If the value is true, it indicates that the adapter will retry the connection to EIS if it cannot connect at startup. The values for the following properties have to be specified: • "Maximum number of retries in case of system connection failure" on page 259 • "Time between retries in case of system connection failure (milliseconds)" on page 265 If the value is false, it indicates that the adapter will not retry the connection to EIS if it cannot connect at startup.
Globalized	No
Bidi supported	No

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 151. 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 152. 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 153. RFC trace level details

Required	No
Possible values	`0` - No error `1` - Errors and warnings `2` - Execution path, errors and warnings `3` - Full Execution path, errors and warnings `4` - Execution path, info messages, errors and warnings `6` - Full execution path, info messages, errors and warnings `7` - Debug messages, full execution path, info messages, errors and warnings `8` - Verbose debug messages, full execution path, info messages, errors and warnings
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

Table 153. RFC trace level details (continued)

Bidi supported	No

RFC trace on

This property specifies whether to generate a text file detailing the RFC activity for each event listener.

Table 154. 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.
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 155. 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 156. 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 157. 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 158. 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 159. Secure Network Connection security level details

Required	Yes, if SncMode is set to 1; no otherwise.

Table 159. Secure Network Connection security level details (continued)

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 160. 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.

If the "Retry EIS connection on startup" on page 261 property is set to true it also indicates the time interval that the adapter will wait in between attempts to retry the inbound connection to the EIS if it cannot connect at startup.

Table 161. Time between retries in case of system connection failure details

Required	No
Possible Values	Positive Integers
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 162. 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 163. 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 164. 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 JNDI, in theWebSphere 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 http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/topic/com.ibm.wsadapters.jca.sap.doc/shared/rsha_in_interpret_prop_details.html.

Table 165. Activation specification properties for ALE inbound processing

Property name		
In the wizard	**In the administrative console**	**Description**
"Failure code" on page 269	aleFailureCode	Specifies the status code for dispatch failure.
"Failure text" on page 270	aleFailureText	Specifies the descriptive text for dispatch failure.
	alePacketUpdate	Specifies if the adapter should send ALEAUD per IDoc or per packet (TID)
"Selective update" on page 270	aleSelectiveUpdate	Specifies which IDoc Type and MessageType combinations are to be updated when the adapter is configured to update a standard SAP status code.
"Status message code" on page 271	aleStatusMsgCode	If required, specifies the message code to use when the adapter posts the ALEAUD Message IDoc (ALEAUD01).
"Success code" on page 271	aleSuccessCode	Specifies the success status code for Application Document Posted.
"Success text" on page 272	aleSuccessText	Specifies the descriptive text for successful Application Document Posted.
"ALE update status" on page 273	aleUpdateStatus	Specifies whether an audit trail is required for all message types.
"Assured once-only delivery " on page 273	AssuredOnceDelivery	Specifies whether to provide assured-once delivery for inbound events.
"Auto create event table" on page 274	EP_CreateTable	Indicates whether the adapter should create the event recovery table automatically if it does not already exist.
"Client" on page 274	Client	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 274	Codepage	Indicates the numeric identifier of the code page.
"Event recovery data source (JNDI) name" on page 275	EP_SchemaName	The schema used for automatically creating the event recovery table.

Table 165. Activation specification properties for ALE inbound processing (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
"Enable Secure Network Connection" on page 272	SncMode	Indicates whether secure network connection mode is used.
"Event recovery data source (JNDI) name" on page 275	EP_DataSource_JNDIName	The JNDI name of the data source configured for event recovery.
"Event recovery table name" on page 276	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 277	RfcTracePath	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Gateway host" on page 277	GatewayHost	The host name of the SAP gateway.
"Gateway service" on page 277	GatewayService	The identifier of the gateway on the gateway host that carries out the RFC services.
"Host name" on page 278	ApplicationServerHost	Specifies the IP address or the name of the application server host that the adapter logs on to.
"IDoc empty tags" on page 278	IDocEmptyTag	Includes empty tags to the unpopulated fields in the IDoc segment, which are sent to a configured endpoint, based on the option selected.
	IgnoreIDocPacketErrors	Determines what the adapter does when it encounters an error while processing the IDoc packet.
"Language code" on page 279	Language code	Specifies the Language code in which the adapter logs on to SAP.
"Logon group name" on page 279	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.
"Load Balancing" on page 279	loadBalancing	Specifies if your SAP configuration uses load balancing
"Maximum Number of retries in case of system connection failure" on page 280	connectionRetryLimit	The adapter will try connecting to the Enterprise Information System (EIS) for a specified number of tries. Select only if you want to reduce the number of connection exceptions in the outbound operation. If selected, adapter will validate the connection for each outbound request.
"Message server host" on page 280	MessageServerHost	Specifies the name of the host on which the message server is running.
"Number of listeners" on page 281	NumberOfListeners	Specifies the number of event listeners that are to be started.
"Partner character set" on page 281	PartnerCharset	Specifies PartnerCharset encoding.
"Password" on page 281	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 282	EP_Password	The user password for connecting to the database.
"Retry EIS connection on startup" on page 282	RetryConnectionOnStartup	Controls whether the adapter retries the connection to the EIS if it cannot connect at startup

Table 165. Activation specification properties for ALE inbound processing (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
"RFC program ID" on page 283	RfcProgramID	The remote function call identifier under which the adapter registers in the SAP gateway.
"RFC trace level" on page 283	RfcTraceLevel	Specifies the global trace level.
"RFC trace on" on page 283	RfcTraceOn	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP system ID" on page 284	SAPSystemID	Specifies the system ID of the SAP system for which logon load balancing is allowed.
"Secure Network Connection library path" on page 284	SncLib	Specifies the path to the library that provides the secure network connection service.
"Secure Network Connection name" on page 285	SncMyname	Specifies the name of the secure network connection.
"Secure Network Connection partner" on page 285	SncPartnername	Specifies the name of the secure network connection partner.
"Secure Network Connection security level" on page 285	SncQop	Specifies the level of security for the secure network connection.
"System number" on page 286	SystemNumber	The system number of the SAP application server.
"Time between retries in case of system connection failure (milliseconds)" on page 286	connectionRetryInterval	Specifies the time interval between attempts to restart the event listeners.
"Trim ALE Idoc field data" on page 286	trimAleData	Specifies if the leading white spaces are to be trimmed by the adapter before sending it to endpoint.
"User name" on page 287	userName	The user account for the adapter on the SAP server.
"User name used to connect to event data source" on page 287	EP_UserName	The user name for connecting to the database.
"X509 certificate" on page 287	X509cert	Specifies the X509 certificate to be used as the logon ticket.

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 166. ALE failure code details

Required	Yes if AleUpdateStatus is set to `True`; no otherwise
Possible values	68 58
Default	40, 51, 68
Property type	Integer

Table 166. ALE failure code details (continued)

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

Failure text

The text that displays in the event that an IDoc is not successfully sent to the endpoint.

Table 167. ALE failure text details

Required	Yes if AleUpdateStatus is set to `True`; no otherwise.
Possible values	`40, 51, 68`
Default	68 Error - no further processing. The values in the text boxes change in accordance with the failure codes.
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

Selective update

Specifies which IDoc Type and MessageType combinations are to be updated.

Table 168. ALE selective update details

Required	No
Default	No default value
Property type	String

Table 168. ALE selective update details (continued)

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

Status message code

This property specifies the message code to use when the adapter posts the ALEAUD01 IDoc with message type ALEAUD.

Table 169. 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

Success code

ALE success code for the successful posting of an IDoc.

Table 170. ALE success code details

Required	Yes if AleUpdateStatus is set to `True`; no otherwise
Possible values	`30, 41, 55`
Default	55 - Application document posted. The values in the text boxes change in accordance with the success codes
Property type	Integer

Table 170. 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

Enable Secure Network Connection

This property indicates whether secure network connection mode is enabled.

Table 171. 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: • SncLib • SncMyname • SncPartnername • SncQop.
Globalized	No
Bidi supported	No

Success text

Indicates the text that displays when an application document is posted successfully.

Table 172. ALE success text details

Required	Yes if AleUpdateStatus is set to `True`; no otherwise.
Possible values	`30, 41, 55`
Default	55 - Application document posted. The values in the text boxes change in accordance with the success codes
Property type	String

Table 172. ALE success text details (continued)

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 173. ALE update status details

Required	Yes
Possible values	`True` `False`
Default	`False`
Property type	Boolean
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 174. Assured once-only delivery details

Required	No
Default	`True`
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.

Table 174. Assured once-only delivery details (continued)

Globalized	No
Bidi supported	No

Auto create event table

Determines if the event table is created automatically.

Table 175. 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.
Globalized	No
Bidi supported	No

Client

This property is the client number of the SAP system to which the adapter connects.

Table 176. 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 177. Codepage number details

Required	No

Table 177. Codepage number details (continued)

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 178. 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

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 179. 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 administrative console. The adapter utilizes data source for *persisting* the event state.
Example	`jdbc/DB2`

Table 179. Event recovery data source (JNDI) name details (continued)

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 180. 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 181. 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 182. 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. This field cannot be edited if you are modifying existing artifacts
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 183. 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 184. Gateway service details

Required	Yes
Default	`sapgw00`
Property type	String

Table 184. Gateway service details (continued)

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 185. 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

IDoc empty tags

This property includes empty tags to the unpopulated fields in the IDoc segment, which are sent to a configured endpoint, based on the option selected.

Table 186. IDoc empty tags

Required	No

Possible Values	`BEFORE_AND_AFTER`
	
	`ONLY_BEFORE`
	
	`ALL_UNPOPULATED_FIELDS_SEGMENTS`

Default	`ONLY_BEFORE`
Property type	String
Usage	Use this property to select the following IDoc empty tag options: • `BEFORE_AND_AFTER` - Include empty tags to the unpopulated fields before and after the populated fields within the IDoc segments. • `ONLY_BEFORE` - Include empty data for the unpopulated fields within an IDoc segment before the populated field. • `ALL_UNPOPULATED_FIELDS_SEGMENTS` - Include empty tags to the unpopulated fields in all the IDoc segments.
Globalized	No
Bidi supported	No

Language code

This property specifies the Language code in which the adapter logs on.

Table 187. 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 188. 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

Load Balancing

This property specifies if your SAP configuration uses load balancing

Table 189. Load balancing details

Required	Yes

Table 189. Load balancing details (continued)

Possible values	`TrueFalse`
Default	False
Property type	Boolean
Usage	This value should be set to true if the SAP configuration uses load balancing. If set to true, Message server host, Logon group and SAP System ID need to be specified.
Globalized	No
Bidi supported	No

Maximum Number of retries in case of system connection failure

This property specifies the number of times the adapter attempts to create a connection to the Enterprise Information System (EIS). The adapter will try connecting to the EIS for the specified number of times. Select only if you want to reduce the number of connection exceptions in the outbound operation. If selected, the adapter will validate the connection for each outbound request.

Table 190. Reset Client details

Required	No
Possible values	Integers
Default	`0`
Property type	Integer
Usage	Only positive values are valid. When the adapter encounters an error related to the outbound connection, it retries to establish a physical connection (when physical connection is not established) for the number of times specified for this property with a time delay specified in the property "Time between retries in case of system connection failure (milliseconds)" on page 230. If the value is `0`, the adapter does not perform any EIS connection validation and executes the outbound operation. if the value is `> 0`, then during each request the adapter validates if the EIS connection is active. • If the connection is valid the operation is completed. • if connection is invalid, the adapter invalidates the current managed connection and a new managed connection is created (new physical connection)
Globalized	No
Bidi supported	No

Message server host

This property specifies the name of the host on which the message server is running.

Table 191. Message server host details

Required	Yes (if load balancing is used)
Default	No default value
Property type	String

Table 191. Message server host details (continued)

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 192. 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. **Note:** The adapter will not start if the number of listeners is 0
Globalized	No
Bidi supported	No

Partner character set

This property specifies the partner character set encoding.

Table 193. 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 194. Password details

Required	Yes
Default	No default value
Property type	String

Table 194. 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

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 195. 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

Retry EIS connection on startup

This property controls whether the adapter retries the connection to the EIS if it cannot connect at startup. This property is used in conjunction with “Maximum Number of retries in case of system connection failure” on page 280 and “Time between retries in case of system connection failure (milliseconds)” on page 286.

Table 196. Retry EIS connection on startup

Required	No

Possible Values	`True`
	
	`False`

Default	`False`
Property type	Boolean
Usage	If the value is true, it indicates that the adapter will retry the connection to EIS if it cannot connect at startup. The values for the following properties have to be specified: • “Maximum Number of retries in case of system connection failure” on page 280 • “Time between retries in case of system connection failure (milliseconds)” on page 286 If the value is false, it indicates that the adapter will not retry the connection to EIS if it cannot connect at startup.
Globalized	No
Bidi supported	No

RFC program ID

This property is the program identifier under which the adapter registers in the SAP gateway.

Table 197. 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 198. RFC trace level details

Required	No
Possible values	`0` - No error `1` - Errors and warnings `2` - Execution path, errors and warnings `3` - Full Execution path, errors and warnings `4` - Execution path, info messages, errors and warnings `6` - Full execution path, info messages, errors and warnings `7` - Debug messages, full execution path, info messages, errors and warnings `8` - Verbose debug messages, full execution path, info messages, errors and warnings
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 199. RFC trace on details

Required	No
Possible values	`True` `False`

Table 199. RFC trace on details (continued)

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.
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 200. 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 201. 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 202. 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 203. 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 204. 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 205. 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 connect to the Enterprise Information System (EIS).

Table 206. Time between retries in case of system connection failure details

Required	No
Possible Values	Positive Integers
Default	`60000`
Unit of measure	Milliseconds
Property type	Integer
Usage	When the adapter encounters an error related to the outbound connection, this property specifies the time interval that the adapter waits in between attempts to reestablish an outbound connection. It is disabled by default and is only enabled when the value of "Maximum Number of retries in case of system connection failure" on page 223 is greater than 0.
Globalized	No
Bidi supported	No

Trim ALE Idoc field data

This property specifies if the leading white spaces are trimmed by the adapter before sending it to endpoint.

Table 207. Trim ALE Idoc field data

Required	No

Possible Values	`True`
	
	`False`

Default	`True`
Property type	Boolean
Usage	Set the value to `True`, if you want the leading white spaces to be trimmed by the adapter before sending it to endpoint. By default, the value is set to `True`. Set the value to `False`, if you do not want the leading white spaces to be trimmed by the adapter.

Table 207. Trim ALE Idoc field data (continued)

Globalized	No
Bidi supported	No

User name

This property is the user account for the adapter on the SAP server.

Table 208. 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 209. 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 210. 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 210. X509 certificate details (continued)

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 JNDI in 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 http://publib.boulder.ibm.com/infocenter/dmndhelp/v7r5mx/topic/com.ibm.wsadapters.jca.sap.doc/shared/rsha_in_interpret_prop_details.html.

Table 211. Activation specification properties for Advanced event processing

Property name		
In enterprise service wizard	**In administrative console**	**Purpose**
"Assured once-only delivery" on page 290	AssuredOnceDelivery	Specifies whether to provide assured-once delivery for inbound events.
"Client" on page 291	Client	The client number of the SAP system to which the adapter connects.
"Codepage number" on page 291	Codepage	Indicates the numeric identifier of the code page.
"Enable Secure Network Connection" on page 292	SncMode	Indicates whether secure network connection mode is used.
"Delivery type (DeliveryType)" on page 292	DeliveryType	Determines the order in which events are delivered by the adapter to the export.
"Event types to process (EventTypeFilter)" on page 292	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 293	RfcTracePath	Sets the fully qualified local path to the folder into which the RFC trace files are written.
"Gateway host" on page 294	GatewayHost	The host name of the SAP gateway.

Table 211. Activation specification properties for Advanced event processing (continued)

Property name		
In enterprise service wizard	**In administrative console**	**Purpose**
"Gateway service" on page 294	GatewayService	The identifier of the gateway on the gateway host that carries out the RFC services.
"Host name" on page 294	ApplicationServerHost	Specifies the IP address or the name of the application server host that the adapter logs on to.
"IDoc empty tags" on page 295	IDocEmptyTag	Includes empty tags to the unpopulated fields in the IDoc segment, which are sent to a configured endpoint, based on the option selected.
"Language code" on page 295	Language code	Specifies the Language code in which the adapter logs on to SAP.
"Logon group name" on page 296	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.
"Load Balancing" on page 296	loadBalancing	Specifies if your SAP configuration uses load balancing
"Maximum number of events collected during each poll" on page 296	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 297	RetryLimit	The number of times SAP JCo tries to restart its server after an error.
"Message server host" on page 297	MessageServerHost	Specifies the name of the host on which the message server is running.
"Partner character set" on page 297	PartnerCharset	Specifies PartnerCharset encoding.
"Password" on page 298	Password	The password of the user account of the adapter on the SAP application server.
"Retry EIS connection on startup" on page 298	RetryConnectionOnStartup	Controls whether the adapter retries the connection to the EIS if it cannot connect at startup
"RFC trace level" on page 299	RfcTraceLevel	Specifies the global trace level.
"RFC trace on" on page 299	RfcTraceOn	Specifies whether to generate a text file detailing the RFC activity for each event listener.
"SAP system ID" on page 300	SAPSystemID	Specifies the system ID of the SAP system for which logon load balancing is allowed.
"Secure Network Connection library path" on page 300	SncLib	Specifies the path to the library that provides the secure network connection service.
"Secure Network Connection name" on page 301	SncMyname	Specifies the name of the secure network connection.
"Secure Network Connection partner" on page 301	SncPartnername	Specifies the name of the secure network connection partner.
"Secure Network Connection security level" on page 302	SncQop	Specifies the level of security for the secure network connection.

Table 211. Activation specification properties for Advanced event processing (continued)

Property name		
In enterprise service wizard	**In administrative console**	**Purpose**
"Stop the adapter when an error is encountered while polling (StopPollingOnError)" on page 302	StopPollingOnError	Specifies whether the adapter stops polling for events when it encounters an error during polling.
"System number" on page 302	SystemNumber	The system number of the SAP application server.
"Time between polling for events (milliseconds)" on page 303	PollPeriod	The length of time that the adapter waits between polling periods
"Time between retries in case of system connection failure (milliseconds)" on page 303	RetryInterval	This property is used by the SAP JCo server for the number of retry attempts made.
"User name" on page 304	userName	The user account for the adapter on the SAP server.
"X509 certificate" on page 304	X509cert	Specifies the X509 certificate to be used as the logon ticket.

Assured once-only delivery

This property specifies whether to provide assured once-only delivery for inbound events.

Table 212. Assured once-only delivery details

Required	Yes
Default	`True`
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

Client

This property is the client number of the SAP system to which the adapter connects.

Table 213. Client details

Required	Yes
Possible values	You can enter a range of values from `000` to `999`.
Default	`100`
Property type	Integer

Table 213. Client details (continued)

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

Client

This property is the client number of the SAP system to which the adapter connects.

Table 214. 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 215. 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

Delivery type (DeliveryType)

This property specifies the order in which events are delivered by the adapter to the export.

Table 216. 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 217. 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: • SncLib • SncMyname • SncPartnername • SncQop
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 218. Event types to process details

Required	No
Possible values	A comma-delimited (,) list of business object types
Default	`null`
Property type	String

Table 218. Event types to process details (continued)

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 events related to the Customer and Order business objects only, 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 219. 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 220. Folder for RFC trace files details

Required	No
Default	No default value
Property type	String

Table 220. Folder for RFC trace files details (continued)

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. This field cannot be edited if you are modifying existing artifacts
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 221. 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 222. 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 223. 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

IDoc empty tags

This property includes empty tags to the unpopulated fields in the IDoc segment, which are sent to a configured endpoint, based on the option selected.

Table 224. IDoc empty tags

Required	No

Possible Values	`BEFORE_AND_AFTER`
	
	`ONLY_POPULATED_FIELDS`

Default	`ONLY_POPULATED_FIELDS`
Property type	String
Usage	Use this property to select the following IDoc empty tag options: • `BEFORE_AND_AFTER` - Include empty tags to the unpopulated fields before and after the populated fields within the IDoc segments. • `ONLY_POPULATED_FIELDS` - Do not include any empty tags to the unpopulated fields.
Globalized	No
Bidi supported	No

Language code

This property specifies the Language code in which the adapter logs on.

Table 225. 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)`.

Table 225. Language code details (continued)

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 226. 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

Load Balancing

This property specifies if your SAP configuration uses load balancing

Table 227. Load balancing details

Required	Yes
Possible values	`TrueFalse`
Default	False
Property type	Boolean
Usage	This value should be set to true if the SAP configuration uses load balancing. If set to true, Message server host, Logon group and SAP System ID need to be specified.
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 228. Maximum number of events collected during each poll details

Required	Yes
Default	`10`

Table 228. Maximum number of events collected during each poll details (continued)

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 SAP JCo tries to restart the server. If the Retry EIS connection on startup property is set to true it also indicates the maximum number of times the adapter will retry the inbound connection to the EIS if it cannot connect at startup.

Table 229. Maximum number of retries in case of system connection failure details

Required	No
Possible values	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 SAP JCo tries to restart the server.
Globalized	No
Bidi supported	No

Message server host

This property specifies the name of the host on which the message server is running.

Table 230. 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 231. Partner character set details

Required	No

Table 231. Partner character set details (continued)

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 232. 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

This property controls whether the adapter retries the connection to the EIS if it cannot connect at startup. This property is used in conjunction with "Maximum number of retries in case of system connection failure" on page 297 and "Time between retries in case of system connection failure (milliseconds)" on page 303.

Table 233. Retry EIS connection on startup

Required	No

Possible Values	`True`
	
	`False`

Default	`False`
Property type	Boolean
Usage	If the value is true, it indicates that the adapter will retry the connection to EIS if it cannot connect at startup. The values for the following properties have to be specified: • "Maximum number of retries in case of system connection failure" on page 297 • "Time between retries in case of system connection failure (milliseconds)" on page 303 If the value is false, it indicates that the adapter will not retry the connection to EIS if it cannot connect at startup.
Globalized	No

Table 233. Retry EIS connection on startup (continued)

Bidi supported	No

RFC trace level

This property specifies the global trace level.

Table 234. RFC trace level details

Required	No
Possible values	`0` - No error `1` - Errors and warnings `2` - Execution path, errors and warnings `3` - Full Execution path, errors and warnings `4` - Execution path, info messages, errors and warnings `6` - Full execution path, info messages, errors and warnings `7` - Debug messages, full execution path, info messages, errors and warnings `8` - Verbose debug messages, full execution path, info messages, errors and warnings
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 235. 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.
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

Table 235. RFC trace on details (continued)

Bidi supported	No

SAP system ID

This property specifies the system ID of the SAP system for which logon load balancing is allowed.

Table 236. 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 237. 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 library path

This property specifies the path to the library that provides the secure network connection service.

Table 238. 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 239. 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 240. 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 241. 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

Secure Network Connection security level

This property specifies the level of security for the secure network connection.

Table 242. 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 243. 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 244. 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.

Table 244. System number details (continued)

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 245. 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
Bidi supported	No

Time between retries in case of system connection failure (milliseconds)

This property is used by the SAP JCo server for the number of retry attempts made. If the "Retry EIS connection on startup" on page 298 property is set to true it also indicates the time interval that the adapter will wait in between attempts to retry the inbound connection to the EIS if it cannot connect at startup.

Table 246. Time between retries in case of system connection failure details

Required	No
Possible Values	Positive Integers
Default	`60000`
Unit of measure	Milliseconds
Property type	Integer
Usage	When the adapter encounters an error related to the inbound connection, this property is used by SAP JCo server.
Globalized	No
Bidi supported	No

User name

This property is the user account for the adapter on the SAP server.

Table 247. User name details

Required	Yes
Default	No default value

Table 247. User name details (continued)

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

This property is the user account for the adapter on the SAP server.

Table 248. 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 249. 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

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.

The bidirectional format consists of five attributes. When you set bidirectional properties, you assign values for each of these attributes. The attributes and settings are listed in the following table.

Table 250. 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

Special characters

A

B

C

D

E

F

X

Printed in USA