WebSphere Adapters

Adapter for SAP Software User Guide

Version 6.0

Note
Before using this information, be sure to read the general information in “Notices” on page 105.

7April2006

This edition applies to version 6, release 0, of IBM WebSphere Adapter for SAP Software (product number 5724L79) and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email 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

WebSphere Adapter for SAP Software Version 6.0 User Guide

The IBM® WebSphere® Adapter for SAP Software facilitates the exchange of business objects between SAP systems and J2EE-based programming models.

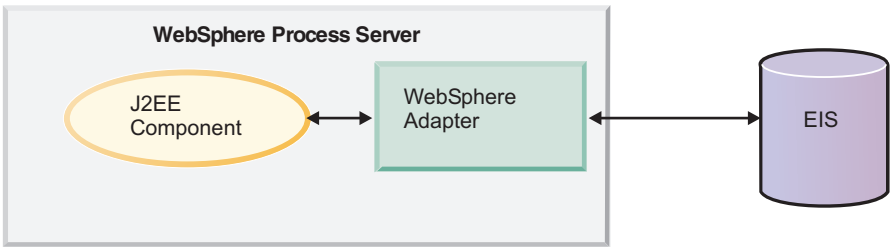
Product overview

This topic introduces the release, product features, and system requirements of the WebSphere Adapter for SAP Software.

IBM WebSphere Adapters

An IBM WebSphere Adapter implements the Java 2 Enterprise Edition (J2EE) Connector architecture (JCA), version 1.5. Also known as resource adapters or JCA adapters, WebSphere Adapters enable managed, bidirectional connectivity between enterprise information systems (EISs) and J2EE components supported by WebSphere Process Server.

A WebSphere Adapter



The IBM(R) WebSphere (R) Adapter portfolio is a new generation of adapters based on the Java 2 Platform, Enterprise Edition (J2EE) standard. JCA is a standard architecture for integrating J2EE applications with enterprise information systems. Each of these systems provides native APIs for identifying a function to call, specifying its input data, and processing its output data. The goal of the JCA is to provide an independent API for coding these functions, to facilitate data sharing, and to integrate J2EE applications with existing and other EISs. The JCA standard accomplishes this by defining a series of contracts that govern interactions between an EIS and J2EE components within an application server.

Fully compliant with the JCA standard, WebSphere Adapters have been developed to run on WebSphere Process Server. A WebSphere Adapter does the following:

- Integrates with WebSphere Process Server.
- Connects an application running on WebSphere Process Server with an EIS.
- Enables data exchange between the application and the EIS.

Each WebSphere Adapter is made up of the following:

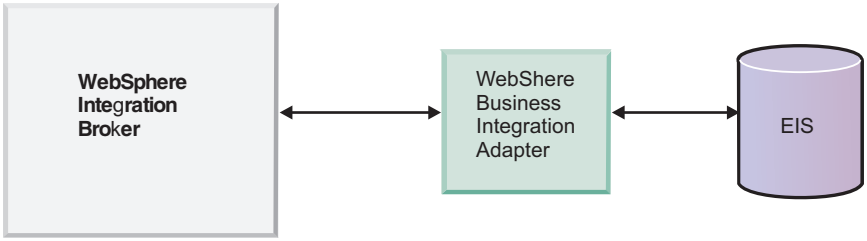
- An implementation of the (J2EE) Connector Architecture (JCA), version 1.5 that supports WebSphere Process Server
- An enterprise metadata discovery component— you use this component with the enterprise service discovery wizard to introspect the EIS— to generate business objects and other service component architecture (SCA) artifacts that are compiled in a standard enterprise application archive (EAR) file.

WebSphere Adapters use service data objects (SDO) for representing data objects.

WebSphere Adapters and WebSphere Business Integration Adapters

Unlike WebSphere Adapters, WebSphere Business Integration Adapters are not JCA-compliant.

A WebSphere Business Integration Adapter



As shown in the figure, WebSphere Business Integration Adapters are distributed. They reside outside of the application server. The server, or integration broker, communicates with this type of adapter through a Java Messaging Service (JMS) transport layer.

Other differences between WebSphere Adapters and WebSphere Business Integration Adapters include the following:

- **Connection management** WebSphere Adapters rely on standard JCA contracts to manage life-cycle tasks such as stopping, starting; WebSphere Business Integration Adapters rely on the WebSphere Adapter Framework to manage connectivity.
- **Event notification** Known as inbound event notification for WebSphere Adapters.
- **Request processing** Known as outbound support in WebSphere Adapters.
- **Object definition** With WebSphere Adapters, you use an enterprise metadata discovery component to probe an EIS and develop business objects and other useful artifacts. This enterprise metadata discovery component is part of the WebSphere Adapter. WebSphere Business Integration Adapters use a separate Object Discovery Agent (ODA) to probe an EIS and generate business object definition schemas.

Audience

The information in this topic defines the users of the WebSphere Adapter products and details the skills they require.

The audience for the adapter user guide includes data and application integrators who are responsible for assembling application components into a complete solution and preparing this solution for testing and deployment. These users require the following general skills:

- A good understanding of the business solution and the business environment
- Knowledge of application and solution components, to enable their efficient collaboration at run-time
- A detailed understanding of databases, data access issues, transactional models and connections across heterogeneous relational databases, queues, and web services
- Familiarity with integration tools

The application integrator is also responsible for detailed testing activities and needs these additional skills:

- Creating required scripts, tools, and templates for testing and deployment
- Creating integration workspaces and integrating systems & subsystems
- Resolving interdependencies between entities such as Enterprise Java Beans (EJBs), workflows, and web pages
- Validating the application or solution

The data integrator is also responsible for enabling access to a range of data sources for the application developers. The required skills include:

- Installing and configuring integration capabilities or point-to-point gateways
- Writing procedures to use database access logic efficiently
- Building data models for external data access tools
- Implementing security measures

Task roadmap: IBM WebSphere Adapter for SAP Software

Perform the tasks listed in the following table to install, configure, deploy, and use the IBM WebSphere Adapter for SAP Software.

Task roadmap

Task	Description
Working with the BAPI interface	This topic describes how the WebSphere Adapter for SAP Software supports the SAP BAPI interface.
Working with the ALE interface	This topic describes how the WebSphere Adapter for SAP Software supports the SAP ALE interface.
Installing the adapter	This topic describes how to install the WebSphere Adapter for SAP Software.
Deploying the adapter	This topic describes how to deploy the WebSphere Adapter for SAP Software.
Configuring the adapter	This topic describes how to configure the WebSphere Adapter for SAP Software.
Troubleshooting the adapter	This topic describes how to troubleshoot the WebSphere Adapter for SAP Software.
Using the sample applications	This topic describes the sample applications for the WebSphere Adapter for SAP Software.

Enterprise service discovery

The enterprise service discovery wizard allows you to generate business objects for enterprise information system (EIS) or database entities.

The enterprise service discovery wizard provides a blue print for business objects. It allows you to browse the metadata information of an EIS or database, enables selection of the artifacts of interest, and generates deployable service objects and descriptions. By selecting meta-object nodes from the metadata tree structure, you can generate business objects for EIS or database entities. The metadata is transformed into service data objects consisting of business graphs and business objects.

The enterprise service discovery wizard allows you to perform the following actions:

- Generate business objects
- Set application-specific information on the business objects
- Set application-specific information on properties
- Provide service descriptions for inbound and outbound events
- Provide connection descriptions for inbound and outbound events

Adapter architecture

The WebSphere Adapter for SAP Software connects to SAP systems running on SAP's web application servers, using the SAP Java™ interface known as SAP Java Connector (SAP JCo). It does so by making calls, modeled as business objects, to the SAP native interfaces and passing data to and from the SAP system. The adapter supports SAP integration interfaces such as Business Application Programming Interface (BAPI) and Application Link Enabling (ALE), as well as RFC-enabled function modules.

The adapter supports outbound processing (from the adapter to the SAP system) and inbound processing (from the SAP system to the adapter) of events.

Outbound event processing

Outbound support allows a client to make calls to the adapter to perform a specific operation in the SAP system. The client requests a connection, which in turn is passed from the adapter to SAP.

Outbound event processing, which the adapter supports for the BAPI and ALE interfaces, consists of the following steps:

1. A Service Component Architecture (SCA) component invokes an interaction with SAP.
2. As a result of the invoked interaction, a business object representing the SAP function call is passed from the component application to the adapter.
3. The adapter extracts the elements from the business object and, using the metadata information from the business object, recognizes the SAP interface to use (BAPI or ALE).
4. Using the SAP Java interface (SAP JCo), the adapter converts the business object data to the appropriate SAP function call.
5. The adapter then executes the function on the destination SAP software system, sending the event data to SAP.

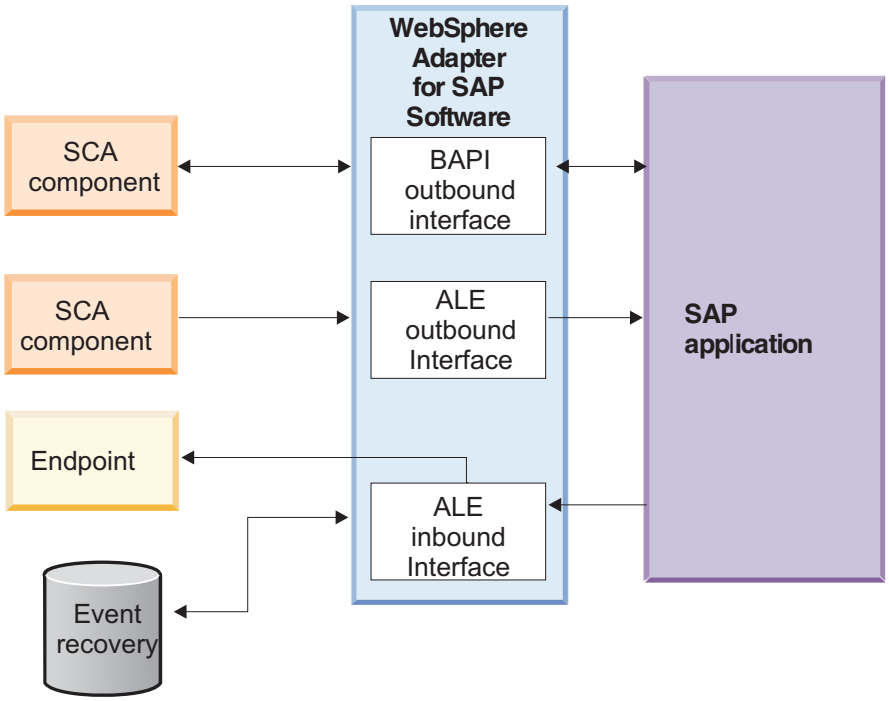
Inbound event processing

Inbound event processing, which the adapter supports for the ALE interface, consists of the following steps:

1. The adapter spawns listener threads to the SAP system.
2. Whenever an event occurs in SAP, the event is pushed to the adapter via the event listeners.
3. The adapter then forwards the event to the endpoint (a message driven bean).
4. The endpoint registers the incoming event.
5. The adapter can track and recover events in case of abrupt termination using the data source for persisting the event state in an event recovery table.

The adapter supports container managed sign-on and basic authentication. It does not support re-authentication.

The following diagram illustrates how the adapter communicates with the SAP system for inbound and outbound processing.



Adapter processing flow

Locale and globalization support

This adapter has been globalized so that it can support single- and double-byte character sets and deliver message text in the specified language.

This adapter supports the processing of bidirectional script data for Arabic and Hebrew languages. To use the bidirectional capacity, you must configure the bidirectional properties. In this user guide, the term *bidirectional properties* refers to the properties that control invocation of bidirectional support.

If your enterprise information system (EIS) uses a bidirectional format that differs from the Windows standard format, all properties with bidirectional support are transformed from the Windows standard format to the bidirectional format of the target EIS. The adapter also transforms such data from the EIS into Windows standard format before passing it to WebSphere Process Server.

The Java(TM) 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). Most components in the WebSphere Business Integration system are written in Java. Therefore, when data is transferred between most WebSphere Business 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 territory, the adapter uses the locale of the system on which it is running.

WebSphere Process Server bidirectional language format

WebSphere Process Server uses the bidirectional language format of ILYNN (implicit, left-to-right, on, off, nominal), which is also the Windows bidirectional language format. All other bidirectional language formats must be converted prior to being introduced to WebSphere Process Server.

Five attributes must be set for the proper bidirectional language format. The attributes and settings are listed in the table titled "Bidirectional attributes."

Bidirectional attributes

Letter position	Purpose	Values	Description	Default setting
1	Order Schema	I or V	Implicit (Logical) or Visual	I
2	Direction	L R C D	Left-to-Right Right-to-Left Contextual Left-to-Right Contextual Right-to-Left	L
3	Symmetric Swapping	Y or N	Symmetric Swapping is on or off	Y
4	Shaping	S N I M F B	Text is shaped Text is not shaped Initial shaping Middle shaping Final shaping Isolated shaping	N
5	Numeric Shaping	H, C, or N	Hindi, Contextual, or Nominal	N

The adapter is responsible for transforming data into a Logical-Left-to-Right format before sending the data into WebSphere Process Server components.

Note: The locale setting of the user interface (browser) defines the bidirectional language display and edit format. WebSphere Process Server user interfaces must convert locale-specific formats to the WebSphere Process Server default format.

Bidirectional property levels

You can set bidirectional properties at several different levels. For more details on these properties and how to set them using the enterprise service discovery wizard, refer to the sections on creating the adapter project and configuring the adapter.

Editing bidirectional properties

You can edit the bidirectional properties for business objects and business object attributes using annotations in the Business Object Editor in WebSphere Integration Developer. The annotations are stored in the business object (the *.xsd file). For

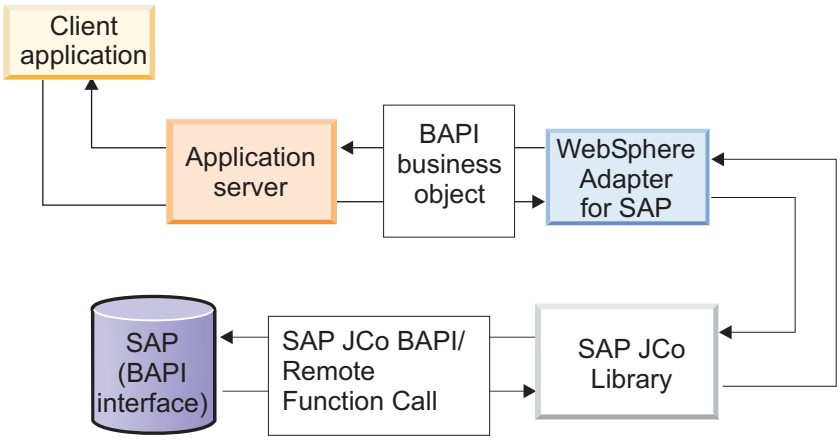
more information, refer to the Business Object Editor documentation on the WebSphere Integration Developer website at http://www.ibm.com/software/integration/wid.

You can also edit certain bidirectional properties once they have been defined by using the assembly editor in WebSphere Integration Developer. For more information on using bidirectional properties at run time, refer to the general technical paper and the adapter technical paper regarding bidirectional support. For more information on the assembly editor, refer to the assembly editor documentation on the WebSphere Integration Developer website at http://www.ibm.com/software/integration/wid.

Working with the BAPI interface

BAPIs are SAP standardized Business Application Programming Interfaces that enable third party systems to interact with SAP systems. These interfaces are implemented as RFC-enabled ABAP functions. The adapter supports outbound processing by modeling SAP BAPI function calls as business objects. These function calls are made to the SAP system, and can create, update, or retrieve data on that system. Because a BAPI is implemented as an RFC-enabled function, the adapter's BAPI interface can support most RFC-enabled functions.

The following diagram illustrates how the adapter handles outbound processing for the BAPI interface.



BAPI outbound processing

The following steps describe how the adapter supports outbound processing for the BAPI interface.

1. The adapter converts a BAPI business object to a SAP JCo function call.
2. The adapter then uses Remote Function Calls (RFCs) in the SAP RFC library to establish an RFC connection to the BAPI interface and executes the corresponding BAPI/RFC function call in the SAP application.
3. After passing the data to SAP, the adapter handles the response from SAP and converts it back into the business object.
4. The business object is then sent to the calling component (JCA client application).

The adapter supports the following kinds of BAPI calls:

- Simple BAPI calls
- BAPI transactions, also called Logical Units of Work

Note that the BAPI/RFC interface supports after-image updates only.

The adapter provides local transaction support for the BAPI interface. The following BAPI calls support local transactions:

- `BAPI_TRANSACTION_COMMIT`
- `BAPI_TRANSACTION_ROLLBACK`

Supporting simple BAPI calls

A simple BAPI call is a synchronous blocking call. The adapter supports simple BAPI calls by representing each with a single business object schema. BAPIs that represent Create, UpdateWithDelete, Retrieve, and Delete operations are grouped with a wrapper business object during business object generation using the enterprise service discovery wizard. The BAPIs 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 runtime in order to execute the simple BAPI call. That is to say, only one BAPI, the one that is associated with the operation to be performed, is called at a time.

The adapter relies on the metadata at the business object and property level to convert the business object into the appropriate SAP API function call. After the SAP system processes the function call and returns it to the adapter, the adapter converts the response into a business object that is stored in the adapter's output record.

The adapter provides local transaction support for the BAPI interface. The following BAPI calls support local transactions:

- `BAPI_TRANSACTION_COMMIT`
- `BAPI_TRANSACTION_ROLLBACK`

For simple BAPIs, BAPI_TRANSACTION_ROLLBACK is called if the BAPI fails and BAPI_TRANSACTION_COMMIT is called if the BAPI succeeds. BAPI transactions are handled differently.

Supporting BAPI transactions

A BAPI transaction, also referred to as BAPI Logical Unit of Work, consists of a set of BAPIs that are executed in sequence so as to complete the entire transaction.

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 transaction. The following three BAPIs illustrate the kind of sequence that forms such a transaction:

- BAPI_ADDRESSEMP_REQUEST
- BAPI_ADDRESSEMP_CHANGE
- BAPI_ADDRESSEMP_APPROVE

The first BAPI in the transaction locks the employee record, the second updates the record, and the third approves the update. The advantage of using a transaction is that the client can invoke the employee record change with a single call, rather

than with three separate calls. In addition, if SAP requires that the BAPIs execute in a specific sequence for the business flow to complete correctly, the transaction supports this sequence.

The adapter supports a BAPI transaction, or sequence of BAPIs, using a top-level wrapper business object that consists of multiple child BAPIs, each one representing a simple BAPI in the sequence. The BAPI transaction wrapper object represents the complete transaction. 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. The adapter uses the application specific information set on the operation to determine the sequence of BAPI calls. BAPI_TRANSACTION_COMMIT can be issued after each BAPI call by specifying the keyword COMMIT in the ASI sequence. The adapter will call BAPI_TRANSACTION_COMMIT at the end of a sequence of BAPI calls even if the application specific information sequence does not specify any COMMITs.

The adapter does not provide an automated rollback mechanism for BAPI transactions. Rollback of a BAPI transaction 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.

Learning about business objects for the BAPI interface

The adapter uses a business object to represent each BAPI it calls in from the SAP system. The structure of the business object varies depending on whether the BAPI is a simple BAPI or a BAPI transaction. BAPIs that represent Create, UpdateWithDelete, Retrieve, and Delete are grouped together into a simple BAPI wrapper.

The adapter depends on the BAPI metadata that is generated by the enterprise service discovery wizard to construct the business objects. This metadata contains BAPI-related information such as the operation of the business object, import parameters, export parameters, table parameters, transaction information, and dependent or grouped BAPIs.

Metadata of BAPI business objects

Business object application-specific information (ASI), a type of metadata, provides the adapter with application-dependent instructions on how to process business objects.

Metadata information instructions are specified at the following levels:

- Business object-level for simple BAPI call objects and BAPI transaction objects
- Operation-level for simple BAPI call objects and BAPI transaction objects
- Property-level for the following:
  - Business object properties that represent child objects
  - Business object properties that represent an array of child objects

The enterprise service discovery wizard automatically generates the appropriate application-specific information (metadata) for each of these elements. The

metadata is in the form of a XSD file, with definitions of the various business object elements (business object, operations, property). It is recommended that you do not change the element names in the generated metadata.

Supported operations and verbs of BAPI business objects

BAPI business objects support certain operations and verbs. An operation reflects the operation to be performed on the business object by the adapter. The verb of the business object reflects its state and is defined at the business graph level for after-image objects only.

Supported operations

The operation of a BAPI business object specifies the BAPI call to execute in the SAP system for that object. The SAP JCo APIs are used to make the call to the SAP software system. The operation is given meaning by the BAPI method. In other words, a BAPI call has inherent functionality, independent of the operation associated with it.

Operations of a business object are invoked by the SCA client component that makes calls to SAP via the adapter.

Then following table defines operations that the adapter supports. Note that these are the expected uses for the operations. The action taken in the SAP application is based on the meaning of the BAPI itself.

Supported Operations

Operation	Definition
After-Image Create	Creates a new entity in SAP that matches the data and structure of the input business object. The business object returned by this operation should accurately reflect the newly created entity in SAP.
After-Image Update with Delete	A special form of the UpdateWithDelete operation that provides better performance. It always requires a ChangeSummary, which is expected to include information about business object-level creations and deletions. This enables the adapter to perform operations without the overhead of retrieving the existing entities from SAP and doing comparisons since the ChangeSummary indicates what needs to be done. If the ChangeSummary is empty, the adapter does not take any action on the request.

Operation	Definition
Retrieve	Rebuilds the complete business object hierarchy. The adapter ensures that the returned hierarchical business object exactly matches the database state of the application entity. Non-key values can be used as criteria. Accepts either an after-image or business object. The comparison in either case will be by equality only. The request business object might contain any of the following: • A top-level business object but no child objects, even though the business object definition includes children • A business object that contains the top-level business object and some of its defined children • A complete hierarchical business object containing all child business objects Retrieve is intended to return a single, unique business object that meets user-defined criteria. The requirement for performing the retrieve is totally dependent on the BAPI. Whatever the BAPI deems as "required" is what allows the retrieve to be successful.
After-Image Delete	Removes an existing entity from SAP and any contained child entities. Note that SAP has the concept of a logical delete, whereby the record is marked as deleted but the entity still exists. This is done in some cases to maintain database integrity because the entity "deleted" is referenced in other entities. Therefore, After-Image Delete behavior depends on the BAPI call.

For an operation that is not supported or does not match with the verb in the business graph, the adapter logs the appropriate error and produces a ResourceException.

Supported verbs

The verb of a BAPI business object specifies the state of the object. The following table lists the verbs that the adapter supports for BAPI business objects.

Supported verbs

Verb	Definition
Create	The top-level business object and all contained children has been created, if this is an inbound event, or should be created in the EIS, if this is an outbound request.

Verb	Definition
UpdateWithDelete	The top-level business object has been or should be modified; this may include addition or deletion of children as well.
Delete	The top-level business object and any contained children have been or should be deleted.

For verbs that are not supported, the adapter produces a ResourceException error.

BAPI business object structure

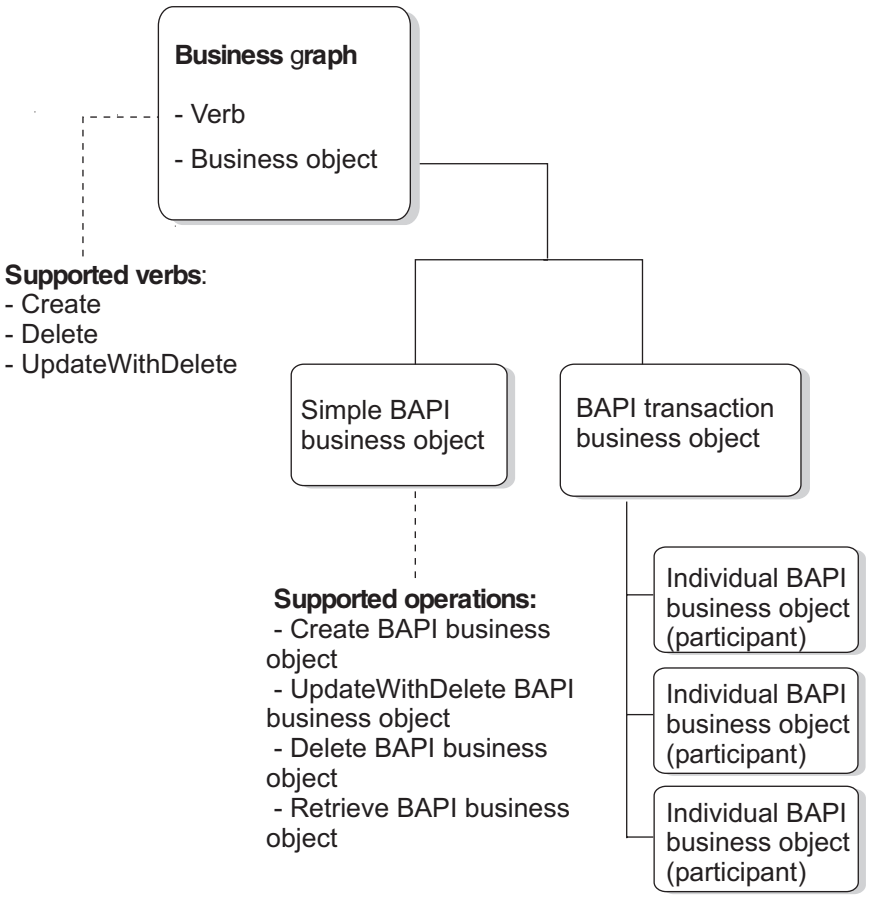
A BAPI business object consists of a business graph with a verb and a business object wrapper containing operations and child objects.

The following table describes BAPI business object elements.

BAPI business object elements

Business object structure element	Description
Business graph	A layer that contains two elements: a verb and a business object. The business graph can refer to a single cardinality business object or a wrapper representing a group of business objects, each with single cardinality.
Verb	BAPI business objects support three verbs: Create, Delete, and UpdateWithDelete. Verbs are at the top level (business object wrapper), not down at the business object level itself.
Business object	The business object itself. It has its own structure, which depends on whether it represents a simple BAPI or a BAPI transaction.

The business graph has the following structure. Note that a business graph has one verb and is for either a simple BAPI or a BAPI transaction (with multiple simple BAPI business object participants).



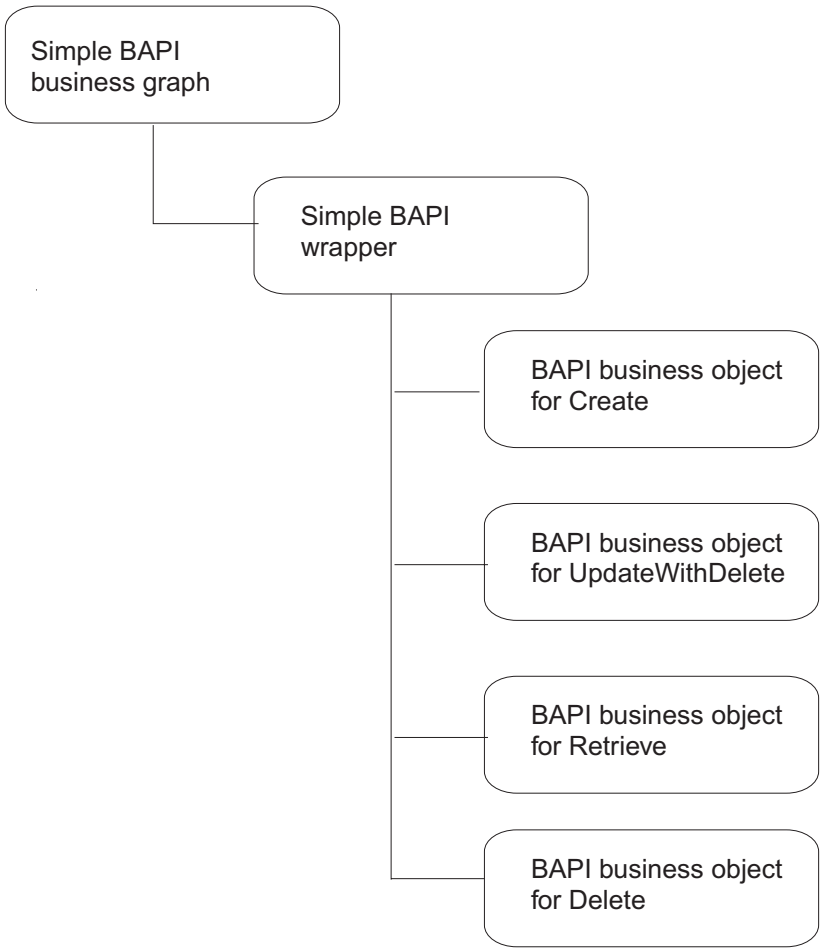
BAPI business graph structure

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 indicates the corresponding parameter. The operation metadata determines the correct BAPI to call.

For a simple BAPI that performs Create, UpdateWithDelete, Retrieve, and Delete operations, each operation is represented by a business object with the business objects being grouped together within a wrapper that is contained in the business graph. Note that the object definition can be associated with multiple operations, but only one operation is executed at runtime. Each business object is a child of the wrapper, and represents a complex property of the wrapper object.

The business object has the following structure.



BAPI business object structure

The wrapper business object, whose name is suffixed with `Wrapper`, is described by metadata. The following table defines the business object property names of a simple BAPI business object.

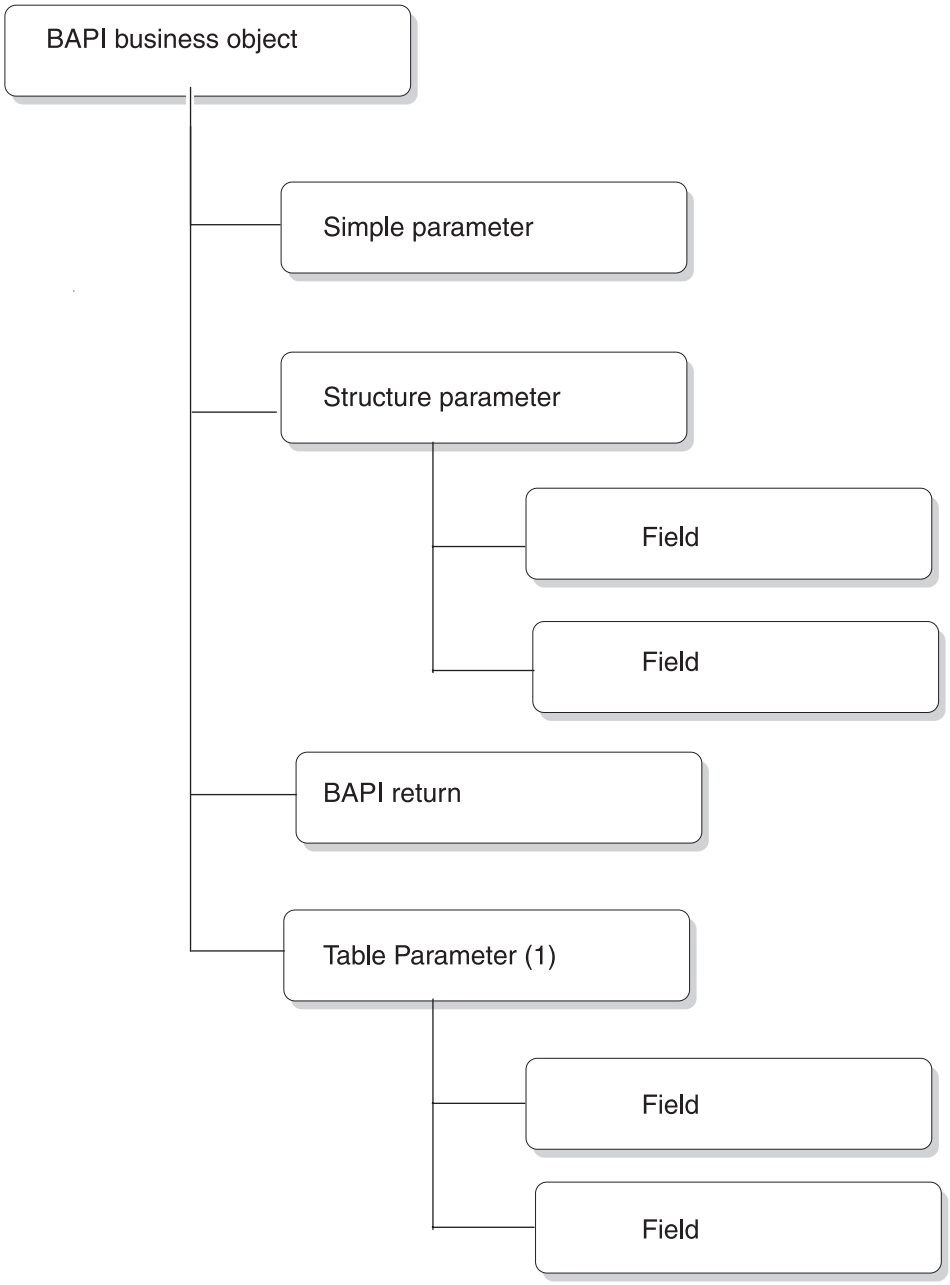
Metadata for wrapper of a simple BAPI business object

Business object property name	Description
Type	The type of the business object. For a simple BAPI the value is set to `BAPI`.
Operation	The valid operations include Create, UpdateWithDelete, and Delete. The specified operation metadata is defined in sapBAPIOperationTypeMetadata and contains the following: • `Name`: Name of the Operation. • `MethodName`: Name of the BAPI associated with the operation.

Child business object structure:

Each BAPI business object (child business object of the wrapper) represents a BAPI call. The business object properties correspond to the parameters of the BAPI call in SAP. These properties are defined by metadata in the business object.

The following diagram illustrates the BAPI child business object structure.



BAPI child business object structure

The adapter supports both single- and multiple-cardinality relationships between business objects. A business object based on a BAPI can contain no more than two levels of hierarchy. Therefore, all BAPI simple parameters correspond to attributes of the top-level business object, and BAPI structure and table parameters correspond to child business objects that contain attributes only.

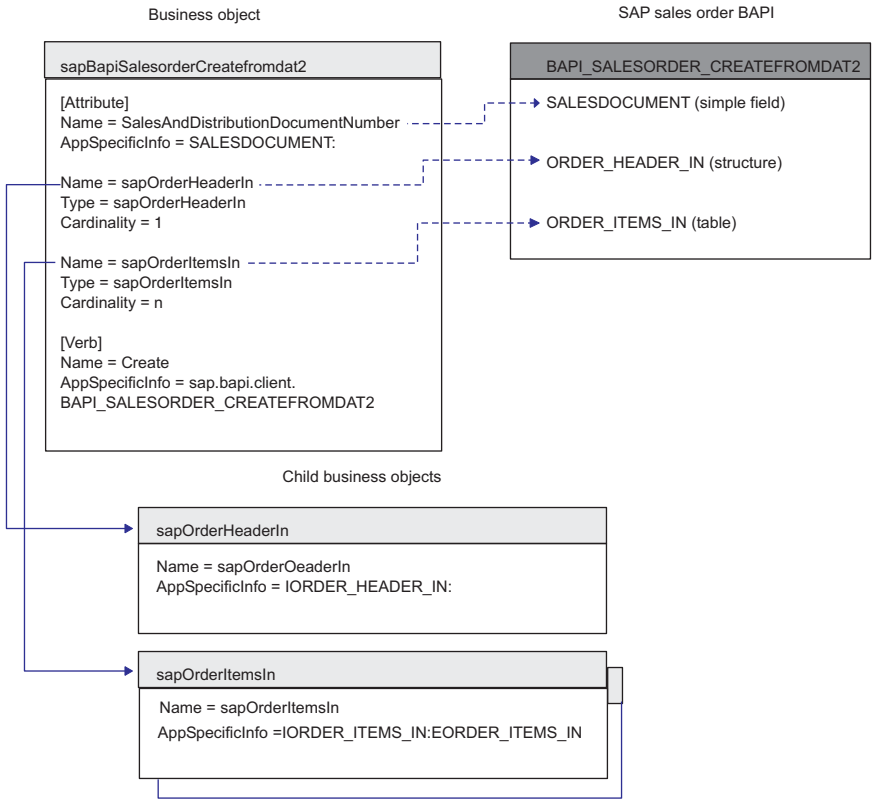
The mapping of the business object structure to the actual BAPI call in SAP is as follows:

Mapping of BAPIs to business objects

BAPI interface parameter	SAP adapter business object property
Simple parameter	A property of the business object. A simple property.
Structure parameter	Single-cardinality child business object. A complex property because it is two-dimensional.
Property	Maps to Field in the business object.
BAPI return	Can contain a structure or a table parameter.
Table parameter	Multiple-cardinality child business object. Three-dimensional.

Import and export parameters can be simple field or structure parameters.

The following figure illustrates the association between a business object and a BAPI. The figure illustrates a fragment of the SapBapiSalesorderCreatefromdat2 business object, which maps to the `BAPI_SALESORDER_CREATEFROMDAT2` BAPI. Notice the child objects `SapOrderHeaderIn` and `SapOrderItemsIn`.



BAPI business object mapping to SAP BAPIs

Property-level metadata:

The adapter uses the value of a property's metadata, or application-specific information, to determine which importing, exporting, and table parameters to use.

BAPI business objects can have the following properties:

- Properties that represent child objects
- Properties that represent an array of child objects

With regard to importing and exporting parameters, the business object property has a metadata element called `ParameterType` that indicates whether the property represents an Import (IN) or Export (OUT) or Import and Export (INOUT) parameter. This identifies the direction of the mapping: adapter business object to SAP BAPI or BAPI to business object. If the value is IN, the property is mapped from the business object to the BAPI. If value is OUT, the property is mapped from the BAPI in the SAP software 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)

The following table describes the application-specific information, or metadata, of a complex property (child) or structure/table property (array of child objects).

Property-level metadata

Metadata name	Description
FieldName	The BAPI field name as represented in SAP.
FieldType	The type of the property as it exists in SAP.
PrimaryKey	A boolean that indicates whether or not this property is a primary key.
ParameterType	Indicates whether this property is an Import (IN) or Export (OUT) or Import and Export (INOUT) parameter.

Operation-level metadata:

The metadata for an operation specifies the method name of the BAPI in the SAP system. This name is used by the adapter to execute the BAPI.

The following table describes the application-specific information, or metadata, of a business object operation.

Operation-level metadata

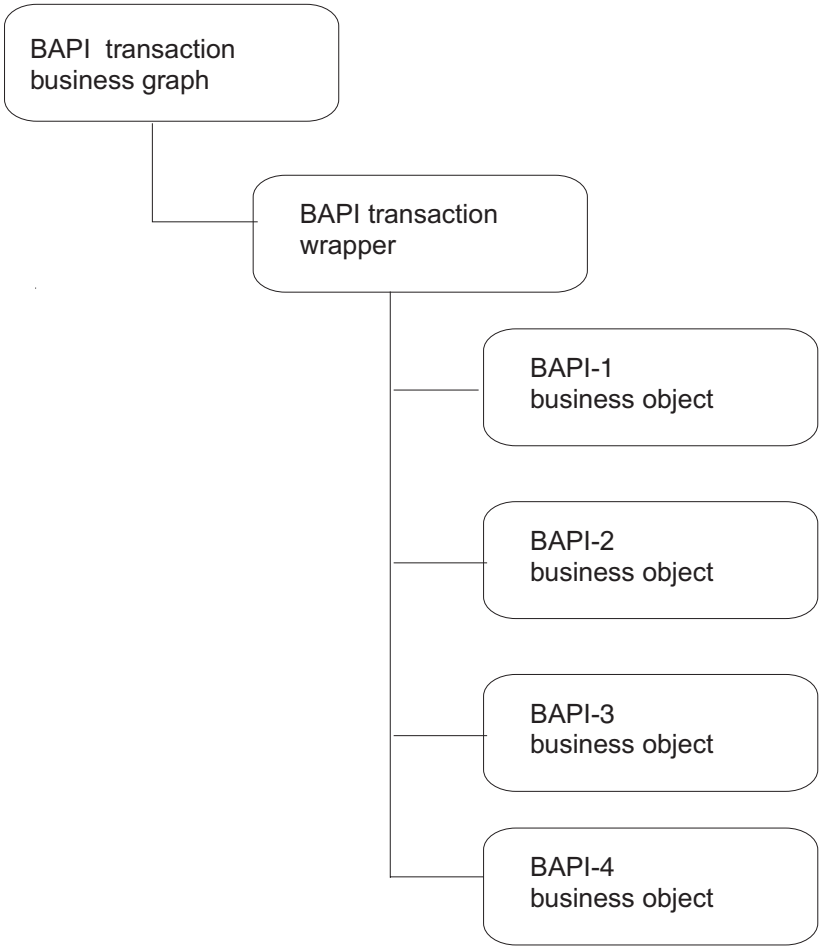
Metadata element	Description
MethodName	The name of the BAPI call (method) in the SAP system.
Name	The name of the business object operation associated with the MethodName.

Business object structure for a BAPI transaction:

A business object representing a BAPI transaction is a wrapper object that contains multiple BAPI objects as children. Each individual child BAPI object within the wrapper BAPI transaction object represents the parameters of a simple BAPI call.

The adapter uses the sequence of operations in the operation metadata to execute the BAPIs in the transaction. In other words, it does not use the order of the child business objects within the business graph.

The following figure illustrates the business object structure of a BAPI transaction.



The following table describes the application-specific information, or metadata, of a BAPI transaction business object.

Metadata for a BAPI transaction business object

Metadata	Description
Type	The business object type. For a BAPI transaction business object, this value is always `BAPITXN`.
Operation	The valid operations include Create, UpdateWithDelete, and Delete. The specified operation is defined in the sapBAPIOperationTypeMetadata tag and contains the following parameters: • `Name`: Name of the Operation • `MethodName`: Name of the BAPI associated with the operation

Naming conventions for BAPI business objects

When you use the enterprise service discovery wizard to generate a business object, a prefix of `Sap` is automatically assigned to the business object name. For child objects, the prefix is prepended to the actual structure or table name. The tool does not permit you to change this prefix.

Naming convention for simple BAPI business objects

The following table describes the naming convention that the enterprise service discovery wizard uses for the business object of a simple BAPI.

Naming convention for simple BAPI business objects

Element	Naming convention
Name of the BusinessGraph	Sap + *Name of the wrapper object as specified by the user in the enterprise service discovery wizard* + Wrapper + BG For example: `SapSalesOrderWrapperBG`
Name of the top-level business object	Sap +*Name of the wrapper object as specified by the user in the enterprise service discovery wizard* + Wrapper For example: `SapSalesOrderWrapper`
Name of the BAPI interface object	Sap + *Name of the BAPI interface* For example: `SapBapiSalesOrderCreateFromDat1` **Note:** The top-level object can contain more than one individual BAPI interface objects
Name of the child object	Sap + *Name of the Structure/Table* For example: `SapReturn` Note that in the case of structures having the same name in different BAPIs or the same structures within a BAPI (for example, one at the export level and one at the table level), the enterprise service discovery wizard generates a unique numeric code and increments to the structure that is existing more than once. For example: `SapReturn619647890`, where 619647890 is the unique identifier suffix appended to the name SapReturn.

Naming convention for BAPI transaction objects

The following table describes the naming convention that the enterprise service discovery wizard uses for BAPI transaction objects.

Naming convention for BAPI transaction objects

Element	Naming convention
Name of the BusinessGraph	Sap + *Name of the wrapper object as specified by the user in the enterprise service discovery wizard* + Txn + BG For example: `SapSalesOrderTxnBG`
Name of the top-level business object	Sap +*Name of the wrapper object as specified by the user in the enterprise service discovery wizard* + Txn For example: `SapSalesOrderTxn`
Name of the BAPI interface object	Sap + *Name of the BAPI interface* For example: `SapBapiSalesOrderCreateFromDat1` **Note:** The top-level object can contain more than one individual BAPI interface objects
Name of the child object	Sap + *Name of the Structure/Table* For example: `SapReturn` Note that in n the case of structures having the same name in different BAPIs or the same structures within a BAPI (for example, one at the export level and one at the table level), the enterprise service discovery wizard generates a unique numeric code and increments to the structure that is existing more than once. For example: `SapReturn619647890`, where 619647890 is the unique identifier suffix appended to the name SapReturn.

Working with the ALE interface

The SAP ALE (Application Link Enabling) interface is part of the integration layer within SAP's Business Framework Architecture (BFA). BFA is a component based architecture that enables business process integration and asynchronous data communication between two or more SAP systems or between SAP and external systems. Application systems are loosely coupled in an ALE integrated system and the data is exchanged asynchronously.

The adapter interacts with the ALE interface to support outbound and inbound processing by enabling the exchange of data in the form of business objects. The data exchange includes the following activities:

- SAP Intermediate Document (IDoc) exchange for inbound and outbound events. The IDocs can be exchanged either as individual documents or in packets.
- Transaction ID (TID) / tRFC (Transactional Remote Function Call) management for inbound events. SAP uses a transaction and its corresponding ID to frame an event, guaranteeing that each piece of data is delivered once and only once from SAP. SAP sends a Transaction ID (TID) with the event data.

Note: Because the adapter uses asynchronous communication, it cannot be used when cross-referencing is required.

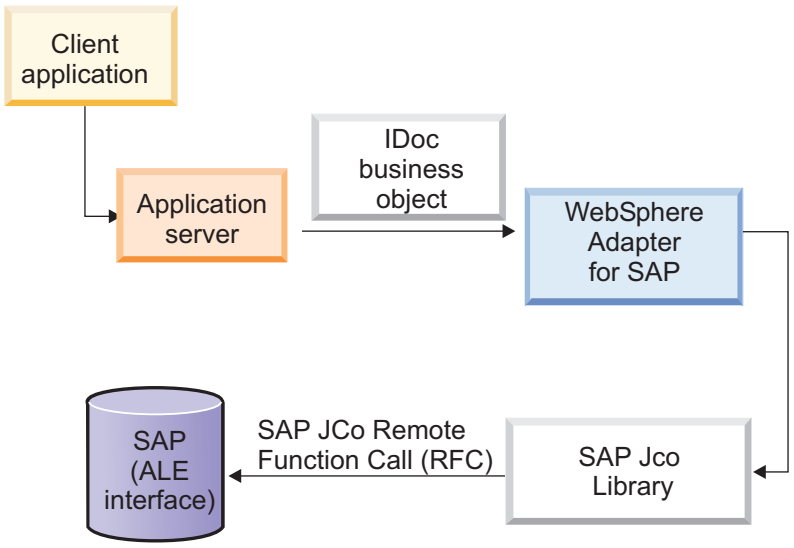
Overview of the ALE interface for outbound processing

The adapter supports ALE outbound event processing by working with business objects that represent outbound IDocs.

The following steps describe how the adapter supports outbound processing for the ALE interface.

1. The adapter uses an IDoc business object to populate the appropriate RFC-enabled function call used by the ALE interface.
2. The adapter establishes an RFC connection to the ALE interface and pass the IDoc data to the SAP system.
3. After passing the data to SAP, the adapter releases the connection to SAP.
4. Because the ALE interface is asynchronous, SAP returns a return code only and a null object to the caller. When no exceptions are raised, the outbound transaction is considered successful. The success of the data being incorporated into the SAP application can be verified by inspecting the IDocs that have been generated in SAP.

Note: The adapter does not provide J2C local transaction support.



ALE outbound processing

If errors exist, the adapter throws the relevant exceptions during ALE outbound processing.

Overview of the ALE interface for inbound processing

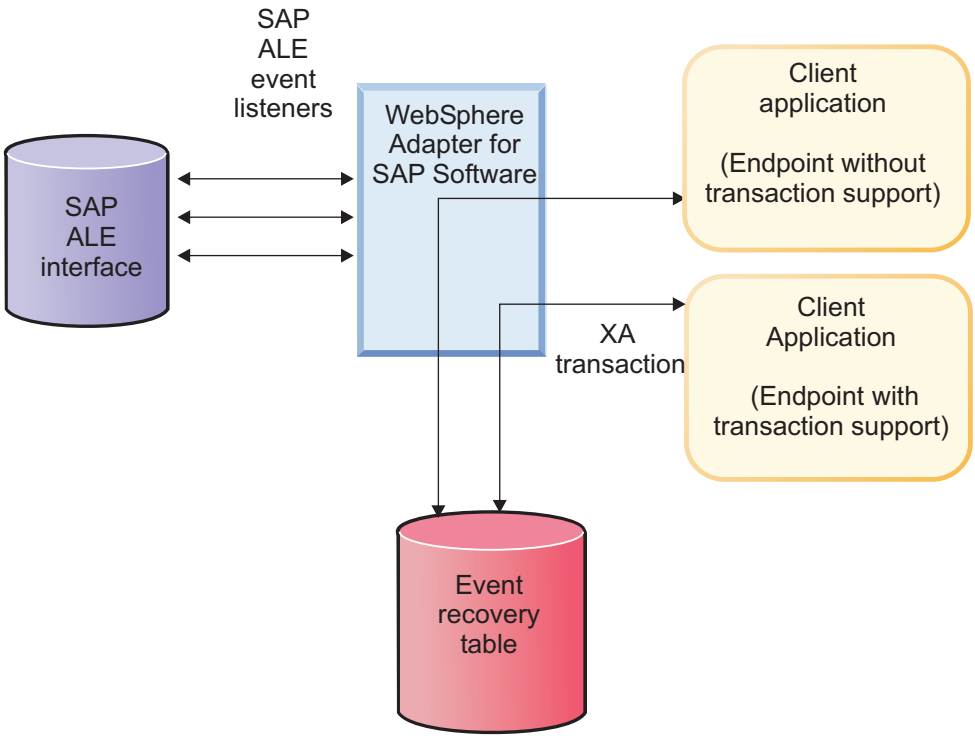
For ALE inbound event processing, the adapter acts as a Remote Function Call (RFC) Server by listening for ALE events from an SAP system. The adapter spawns listener threads to the SAP system. Whenever an event occurs in SAP, the event is pushed to the adapter via the event listeners.

The listener threads listen continuously in a synchronous manner for ALE events coming from the ALE-specific functions that the adapter supports. The threads do the following:

- Use a program identifier to register with the SAP Gateway

- Identify to the SAP Gateway the ALE-specific RFC-enabled functions that they support. These functions are `idoc_inbound_asynchronous` and `inbound_idoc_process`.
- Receive events from the ALE-specific function.

The following diagram illustrates the flow of inbound events between the SAP system and the adapter:



ALE inbound processing

The adapter receives the event from SAP and then converts it into a business object before sending it to the message endpoint on the client application. Note that the adapter can deliver objects to endpoints that support transactions as well as endpoints that do not support transactions:

- For endpoints that support transactions, the adapter delivers the business object as part of a unique XA transaction (a two-phase commit transaction) controlled by the application server. The message endpoint must be configured to support XA transactions.
- For endpoints that do not support transactions, the adapter delivers the business object to the endpoint. The event delivery is guaranteed only when XA transaction processing is supported.
- The adapter supports the tRFC protocol for delivering ALE events from the SAP system to the endpoint (client application).

Although tRFC significantly improves the reliability of the data transfer, 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 listener threads. However, at some point all ALE transactions are transferred. For further details about tRFC, refer to the SAP documentation.

Event recovery

The adapter supports event recovery for ALE inbound processing in case of abrupt termination. During event processing, the adapter persists the event state in an event recovery table that resides on the data source.You must set up the database before you can create the event recovery table. Consult your database administrator for details about creating the database.

Database support

The adapter supports the following databases for event recovery:

- IBM(R) Informix® Dynamic Server, Version 9.40.x
- Cloudscape™ 5.1
- SQL Server Enterprise 2000
- DB2®(R) Enterprise Server Edition 8.2
- Oracle9i Enterprise Edition Release 2

To connect to the various supported databases, do the following:

- Configure adapter properties as described in the following table, "Database connection information." The properties not listed in this table can be added to the EDTExtendedProperties.properties file as name-value pairs.

  Database connection information

Supported database	Configuration property information	Additional information
• IBM(R) Informix Dynamic Server, Version 9.40.x	• EDTDriverName = com.informix.jdbcx. IfxXADataSourc • EDTServerName = *[The name of the Informix instance on the server]* • EDTPortNumber =*[port number]* • EDTDatabaseName = *[The name of the database]*	In the `EDTExtendedProperties. properties` file, set `ifxIFXHOST=`*[The physical name of the database server]*
• Cloudscape 5.1	• EDTDriverName = com.ibm.db2j.jdbc. DB2jXADataSource • EDTDatabaseName = *[any name. Database need not exist]*	
• SQL Server Enterprise 2000	• EDTDriverName = com.microsoft. jdbcx.sqlserver. SQLServerDataSource • EDTServerName = *[name of server hosting SQLServer]* • EDTDatabaseName = *[name of the existing database]* • EDTPortNumber = *[port]*	In the `EDTExtendedProperties. properties` file, set `selectMethod=Cursor`

Supported database	Configuration property information	Additional information
• DB2(R) Enterprise Server Edition 8.2	• EDTDriverName = COM.ibm.db2.jdbc.DB2XADataSource • EDTDatabaseName = *[an existing database]*	
• Oracle9i Enterprise Edition Release 2	• EDTDriverName = oracle.jdbc.xa.client.OracleXADatasource • EDTURL = *[The URL for the DB]*. For example: `jdbc:oracle:thin:@myServer:1521:myDatabase`	

- Copy the file `EDTExtendedProperties.properties` to the following directory on WebSphere Process Server: <*WPS_INSTALL*>\profiles\<*your-profile*>\installedApps\<*host-name*>Node01Cell\<*your-application-name*>App.ear\CWYAP_SAPAdapter.rar\

Event recovery table

The event recovery table contains the following fields:

Event recovery table fields

Table field name	Type	Length	Description
TID	VARCHAR	30	Transaction ID for tRFC (Transactional Remote Function Call) protocol.
Status	VARCHAR	10	Event processing status. Possible values are CREATED, EXECUTED, PARTIAL, and ROLLBACK.
NumIDocs	INT		Total number of IDocs in the packet.
NumIDocsProcessed	INT		Total number of IDocs from the packet that are successfully processed.
CurrIDoc	INT		Sequence number of the IDoc in the packet that the adapter is currently processing

Note: The `AutoCreateEDT` configuration property in the J2C activation specification properties determines whether the event recovery table is created automatically or not. The default value of this property is True (create the

table automatically). For manual creation, use the information provided in the "Event recovery table fields table."

Event processing for a single IDoc

An IDoc corresponds to a single business object. The adapter can process an ALE event that contains just one IDoc.

The following steps describe how the adapter processes an inbound event for a single IDoc.

1. When the SAP system pushes the Transaction ID (TID) to the adapter, the adapter checks the status of the event.
   - If this is a new event, the adapter stores the TID along with a status of `CREATED` in the event recovery table.
   - If the event status is `ROLLBACK`, the adapter updates the status to `CREATED`.
   - If the event status is `EXECUTED`, the adapter returns a return code of `SUCCESS` to the SAP system.
2. The SAP system pushes the single IDoc to the adapter, which parses and converts the IDoc to a business object and stores it in memory.
3. The adapter sends the business object to the appropriate message endpoint. Note that the adapter can deliver objects to endpoints that support transactions as well as endpoints that do not support transactions:
   - For endpoints that support transactions, the adapter delivers the business object as part of a unique XA transaction (a two-phase commit transaction) controlled by the application server. When the endpoint receives the event and the transaction is committed, the status of the event is updated to `EXECUTED`. The message endpoint must be configured to support XA transactions.
   - For endpoints that do not support transactions, the adapter delivers the business object to the endpoint and updates the status of the event to `EXECUTED`.
4. If an exception occurs while either the adapter processes the event or the endpoint receives the object, the adapter throws an exception to the SAP system.
5. The SAP system sends a `ROLLBACK` call to the adapter and the event status is updated to `ROLLBACK`.
6. If no exception occurs, the SAP system sends a `COMMIT` call to the adapter, and the event is updated to `EXECUTED`.
7. The SAP system sends a `CONFIRM` call to the adapter.
8. The adapter deletes the records with an `EXECUTED` status and logs a common event infrastructure (CEI) event that can be used for tracking and auditing purposes.

Event processing for an IDoc packet

An inbound event may contain multiple IDocs, with each IDoc corresponding to a single business object. The multiple IDocs are pushed out by the SAP system to the adapter in the form of an IDoc packet.

Note: The enterprise service discovery wizard allows you to create an ALE wrapper object. The wrapper object will contain a business object, of multiple cardinality, that represents an IDoc. All instances of this business object will be passed into the ALE interface in one RFC-enabled function call.

The following steps describe how the adapter processes an inbound event for an IDoc packet that contains multiple individual IDocs.

1. When the SAP system pushes the Transaction ID (TID) to the adapter, the adapter checks the status of the event.
   - If this a new event, the adapter stores the TID along with a status of `CREATED` in the event recovery table.
   - If the event status is `ROLLBACK`, the adapter updates the status to `CREATED`.
   - If the event status is `EXECUTED`, the adapter returns a return code of `SUCCESS` to the SAP system.
2. The SAP system pushes the IDoc packet to the adapter, which parses and converts the IDoc into multiple business objects and stores them in memory.
3. The adapter also updates the `NumIDocs` column (or table field) in the EventRecovery table to the number of IDocs in the packet. This number is used for audit and recovery purposes. If the adapter encounters an error while processing the IDoc packet, it can behave in one of two different ways, depending on the `IgnoreIDocPacketErrors` configuration property:
   - If the `IgnoreIDocPacketErrors` property is set to false, the adapter stops processing any further IDocs in the packet and reports errors to the SAP system.
   - If the `IgnoreIDocPacketErrors` property is set to true, the adapter log an error and continues processing the rest of the IDocs in the packet. The status of the transaction is marked as `PARTIAL`. In this case, the adapter log shows the IDoc numbers that failed and you must resubmit those individual IDocs separately. You must then manually maintain these records
4. The adapter sends the business objects to the message endpoint, one after the other, and updates the `CurrIDoc` 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.
5. When the endpoint receives the event and the transaction is committed, the adapter increments the number in the `CurrIDoc` property. The message endpoint must be configured to support XA transactions.
6. After the adapter delivers all the business objects in the IDoc packet to the message endpoint, it updates the event status to `EXECUTED`.
7. The SAP system sends a `COMMIT` call to the adapter.
8. 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 `CurrIDoc` property, even if `IgnoreIDocPacketErrors` is set to true. This is required in case the user terminates the adapter manually while processing an IDoc packet.
9. The adapter logs a common event infrastructure (CEI) event that can be used for tracking and auditing purposes.
10. If an exception occurs either while the adapter processes the event or when the endpoint raises an exception, the event status is updated to `ROLLBACK`.
11. The SAP system sends a `CONFIRM` call to the adapter. If the status is `EXECUTED`, the adapter deletes the records from the event recovery table.

Updating the IDoc status

You can configure the adapter to update the IDoc status for the purpose of monitoring your IDoc processing. If the adapter configuration property `ALEUpdateStatus` is set to true (indicating that an audit trail is required for all message types), then the adapter updates the IDoc status of ALE business objects

that are retrieved from the SAP system. The update is accomplished by updating a status IDoc called ALEAUD that the adapter sends to the SAP system as an inbound IDoc event. After the event is sent to the message endpoint, the adapter updates the status of the IDoc in SAP to indicate a failure or success in processing.

The following table defines the IDoc status codes:

IDoc Status codes values

IDoc status code value	Description
12	Dispatch processed without errors.
11	Error during dispatch.

An IDoc that is not successfully sent to the endpoint is considered a failure and the IDoc status is updated by the adapter to 11. Likewise, an IDoc that reaches the endpoint is considered as successfully processed and in this case the status of that IDoc is updated in SAP to 12.

These codes and their associated text are configurable properties of the adapter, as specified in the J2C Activation Specification properties. The following table lists the properties and their values.

Configuration properties for IDoc status codes

Adapter property	Value
ALESuccessCode	12
ALEFailureCode	11
ALESuccessText	Dispatch OK
ALEFailureText	Error during dispatch

For batched IDocs (an IDoc packet), the adapter retrieves either all IDocs with the same Transaction ID (TID) or no IDocs at all. If all IDocs with the same TID are retrieved, then the status code is updated to 12 (all success). If no IDocs are retrieved, then the status code remains as 03.

ALE interface prerequisites

The adapter has certain prerequisites for running with the SAP ALE interface.

To run the adapter with the ALE interface, you must do the following:

- Check the configuration of your SAP system
- Configure SAP to update the IDoc status
- Configure the adapter's J2C activation specification properties

Check the configuration of your SAP system

Before running the adapter with the ALE interface, verify that the SAP system is properly configured to process business objects. The following conditions apply to both inbound and outbound processing:

- Check that the logical systems are defined and assigned for the SAP system and external system (transaction code SALE).

- Check that the distribution model has been maintained, and that the required message types have been added to the model (transaction code BD64).
- Check that there are partner profiles for the logical system or distribution model (transaction code WE20).
- Check that the port definition (transaction code WE21) is defined for the version of IDoc record types that you want.

Configure the adapter's J2C activation specification properties

J2C activation specification properties (also referred to as message endpoint properties), correspond to the ActivationSpec interface of the J2EE Connector Architecture Specification. These properties are relevant to working with the ALE interface because an activation specification is a JavaBean used during endpoint activation. Endpoint activation is the process of notifying the adapter of eligible listener threads. For inbound processing, the adapter uses these listeners to receive events from SAP before forwarding them to the endpoint (a message driven bean).

To enable the adapter to work with the SAP ALE interface, you must configure the J2C activation specification properties.

Configure SAP to update the IDoc status

For inbound processing, you must do the following 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:

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

Learning about business objects for the ALE interface

The adapter represents Intermediate Document (IDoc) data structures as business objects. The enterprise service discovery wizard uses the SAP system's native IDoc definitions as templates for business object definitions when connecting to the ALE interface.

The adapter depends on the IDoc metadata that is generated by the enterprise service discovery wizard to construct the business objects. This metadata contains ALE-related information such as the verb of the business object, import parameters, export parameters, table parameters, transaction information, and single or packaged IDocs.

ALE business object structure

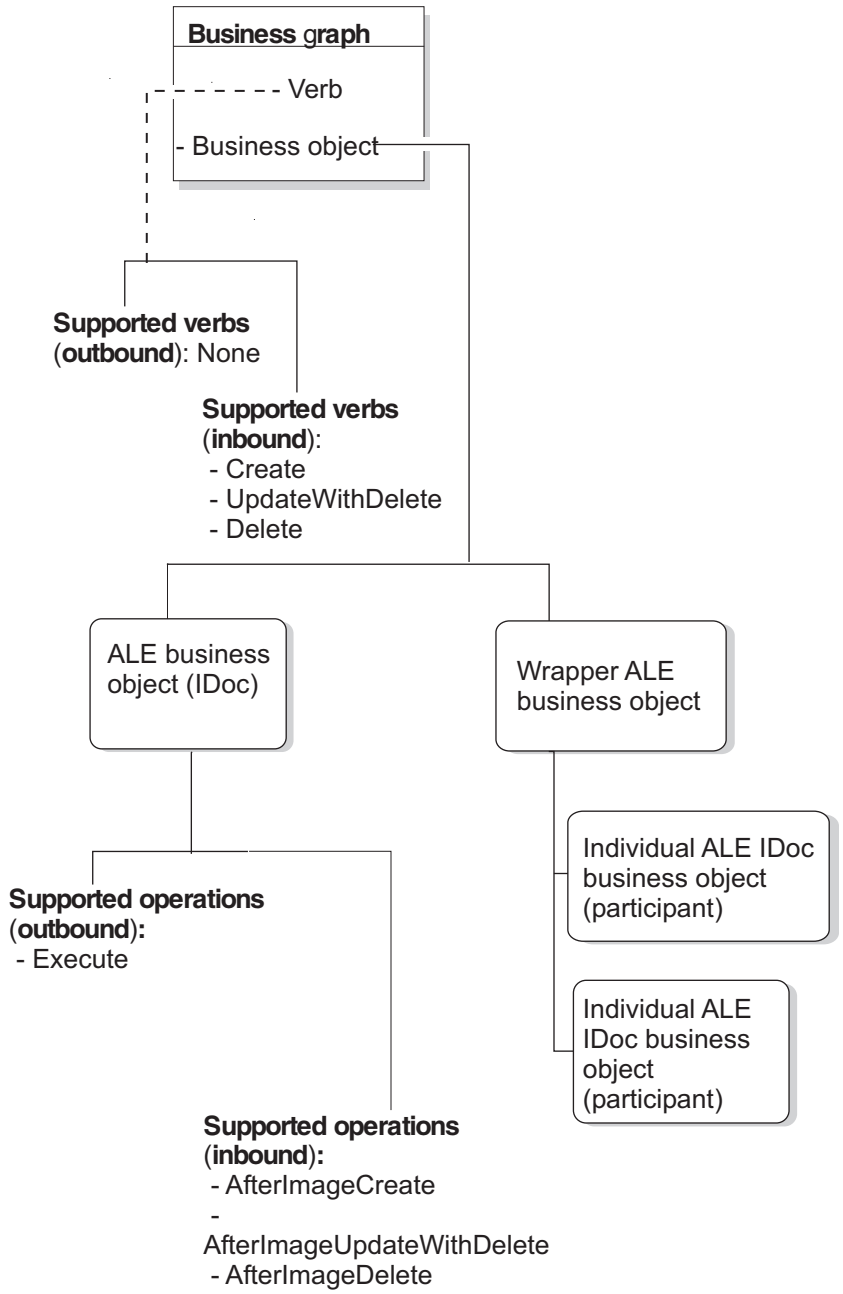
This topic describes the structure of ALE business objects.

The ALE business object structure consists of the following elements:

ALE business object elements

Business object structure element	Description
Business graph	A wrapper that contains two elements: a verb and a business object. The business graph can refer to a single cardinality business object or a wrapper representing a group of business objects, each with single cardinality.
Verb	ALE IDoc business objects support the following verbs: • Outbound business objects: No verb support • Inbound business objects: Create, UpdateWithDelete, Delete
Business object	The business object itself. It has its own structure, which depends on whether it represents a single ALE IDoc or a wrapper ALE business object (which contains multiple single ALE IDocs).

The business graph has the following structure:



ALE business graph structure

Business object structure for a single IDoc:

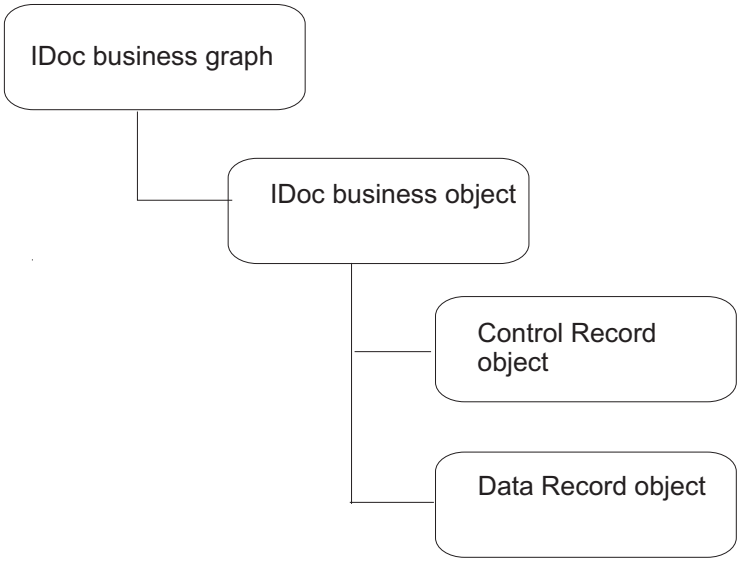
A business object for a ALE IDoc consists of two child objects: a Control Record object and a Data Record object.

The Control Record object, which the enterprise service discovery wizard generates automatically for you, contains the metadata required by the adapter to process the business object.

The Data Record object contains the actual business object data to be processed by the SAP software application, as well as the metadata required by the adapter to

convert the business object to an IDoc structure for the RFC call. The top level of the Data Record business object corresponds to the basic IDoc type. This top-level business object contains an attribute that represents a child business object or an array of child business objects (one for each IDoc segment). The structure and hierarchy of the child business objects match that of the IDoc segments in the basic IDoc type.

The following diagram illustrates the business object structure of a ALE IDoc.

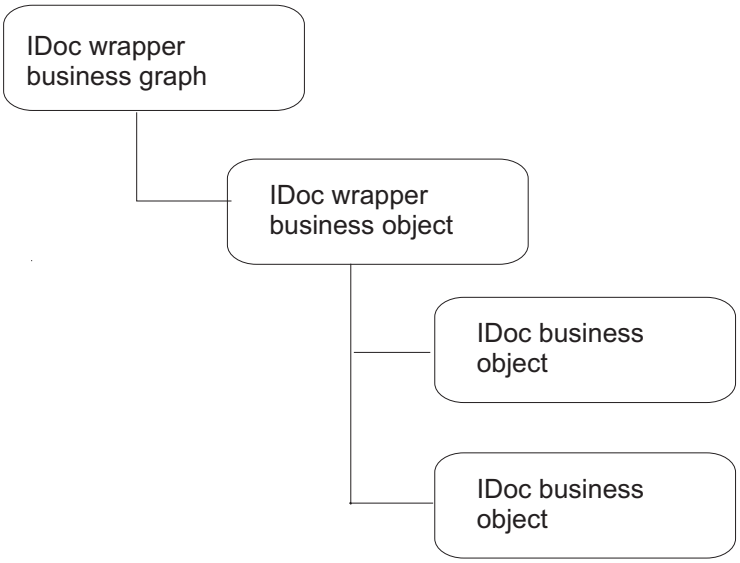


ALE IDoc business object structure

Business object structure for an IDoc packet:

During outbound processing, the adapter can send a business object to the SAP application that represents an IDoc packet. This business object is a top-level wrapper object that contains multiple IDoc child objects, each one corresponding to a single IDoc.

The following diagram illustrates the business object structure of the top-level wrapper object that represents an IDoc packet.



Business object structure of IDoc wrapper object (for an IDoc packet)

Supported verbs and operations of ALE business objects

ALE IDoc business objects support certain verbs and operations. The verb of a business object reflects its state and is definable for after-image objects only. An operation reflects the operation to be performed on the business object by the adapter.

Supported verbs

ALE inbound business objects support the following verbs:

Supported verbs: ALE inbound business objects

Verb	Definition
Create	The top-level business object and all contained children will be created.
UpdateWithDelete	The top-level business object will or should be modified. Can include adding and deleting child objects. Any and all deleted child objects are known and reflected.
Delete	The top-level business object and any contained children will or should be deleted.

Verb determination for inbound processing

The adapter sets the verb property of the business graph for the ALE business object before sending the business object to the endpoint. The verb is determined by comparing the metadata defined in the application specific information of the business object operations with the following IDoc Control Record fields:

- Logical_message_type (MESTYP)
- Logical_message_code (MESCOD)
- Logical_message_function (MESFCT)

Supported verbs: ALE outbound business objects

ALE outbound business objects provide no verb support. The adapter ignores the value in the verb property of the business object graph.

Supported operations

For ALE inbound business objects, the application specific information of an operation is used to set the verb of the business graph. The application specific information of an operation contains the message type, message code and message function for an IDoc type. The business graph verb is set to the verb that corresponds to the operation that has application specific information matching the Control record fields. The adapter supports the following inbound operations.

Supported operations: ALE inbound business objects

Operation	Definition
AfterImageCreate	The top-level business object and all contained children will be created.
AfterImageUpdateWithDelete	The top-level business object will or should be modified. Can include adding and deleting child objects. Any and all deleted child objects are known and reflected.
AfterImageDelete	The top-level business object and any contained children will or should be deleted.

The operation of an ALE outbound business object is invoked by the SCA client application that makes calls to SAP via the adapter. The SCA client must be designed so that the calls made by its InteractionSpec implementation invoke the operations. The adapter supports the following outbound operation.

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. In other words, no response is sent back.

For all other operations, the adapter logs the appropriate error and raises a ResourceException.

Metadata of ALE business objects

Business object application-specific information (ASI), which is a type of metadata, provides the adapter with application-dependent instructions on how to process business objects.

Metadata is specified at the following levels:

- IDoc business object-level (for individual IDocs)
- IDoc wrapper business object-level (for IDoc packets)
- Operation-level for individual IDoc business objects
- Property-level

(Note that there is no metadata at the IDoc Data Record or IDoc Control Record child business object-level.)

The enterprise service discovery wizard automatically generates the appropriate application-specific information (metadata) for each of these elements. The metadata is in the form of an XSD file, with definitions of the various business object elements (business object, operations, property). It is recommended that you do not change the element names in the generated metadata.

Business object-level metadata for ALE business objects:

Business object-level metadata for ALE business objects defines the top-level wrapper of an IDoc.

The following table describes the business object-level metadata of an ALE inbound business object:

Business object-level metadata

Metadata	Description
Type	The business object type. For ALE objects, type is always set to `ALE`.
Operation	Each **inbound** operation contains the following parameters: • `Name`: Name of the operation (`Create`, `UpdateWithDelete`, 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. Each **outbound** operation contains the following single parameter: • `Name`: Name of the operation., which for outbound is always `Execute`.

Operation-level metadata for ALE business objects:

The operation-level metadata, or application-specific information, for an ALE business object specifies the operation that will post the IDoc object to the SAP application. The method name of the RFC call that the operation represents is based on the value that is in TABNAM field of the control record.

The following table describes the application-specific information, or metadata, of an ALE business object operation. Note that outbound objects use only the Name metadata element (MsgType, MsgCode, and MsgFunction are used for inbound objects only).

Operation-level metadata

Metadata element	Description
Name	Name of the operation.
MsgType	The message type configured for the IDoc (for inbound objects only).

Metadata element	Description
MsgCode	The message code configured for the IDoc (for inbound objects only).
MsgFunction	The message function configured for the IDoc (for inbound objects only).

Property-level metadata for ALE business objects:

This topic describes the metadata of ALE business object properties.

The following table describes the property-level application-specific information, or metadata, of a business object.

Property-level metadata

Metadata name	Description
FieldName	Actual IDoc field name as represented in SAP.
SegmentHierarchy	Hierarchy of the segment in the IDoc.
OffSet	The offset value of the current property in the IDoc.
PrimaryKey	A boolean that indicates whether or not this property is a primary key.
ForeignBOKeyref	Used for the DummyKey property to hold the xpath to the primary key on the control or data record business object property, which you set using the business object editor in the enterprise service discovery wizard.

Mapping dummy keys

Mapping of dummy keys allows you 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.

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 xpath of the property from which the value should be set. In other words, the property level application-specific information is set to the xpath within the business object hierarchy of the attribute that is being mapped to the top-level object.
- If multiple cardinality objects are found in this path, the adapter uses the cardinality defined in the xpath. This is true for all multiple cardinality objects wherever they occur in the hierarchy. The following line of code is an example of the xpath: `<sapasi:ForeignBOKeyRef>Orders05/Orders05DataRecord/Orders05E2edk14[1]/OrgID />`
- If the application-specific information is incorrect or if the mapped property value is empty, the adapter fails the event. This is also the case when the application-specific information is configured to set an object type value as the dummyKey. Note that dummyKey can only contain a simple type.

Naming conventions for ALE business objects

When you use the enterprise service discovery wizard to generate a business object, a prefix of `Sap` is automatically assigned to the business object name. The tool does not permit you to change this prefix.

Naming convention of a single ALE IDoc business object

The naming convention of a single ALE IDoc business object is as follows:

prefix + *NameofIDoc* + *[Name of Extension type IDoc]*

Where *[Name of Extension type IDoc]* represents an optional entry. It is included in the name only if the selected IDoc is an Extension Type IDoc.

Naming convention of an ALE IDoc wrapper business object

An IDoc wrapper business object is used for business objects that represent IDoc packets.

The naming convention of an ALE IDoc wrapper business object graph is as follows:

prefix + *NameofIDoc* + *[Name of Extension type IDoc]* + Wrapper + BG

The naming convention of the top level wrapper object of an ALE IDoc packet is as follows:

prefix + *NameofIDoc* + *[Name of Extension type IDoc]* + Wrapper

The naming convention of each participant single IDoc business object is the same as all single IDoc business objects:

prefix + *NameofIDoc* + *[Name of Extension type IDoc]*

Where *[Name of Extension type IDoc]* represents an optional entry. It is included in the name only if the selected IDoc is an Extension Type IDoc.

Note that in case of an IDoc duplicate name, the enterprise service discovery wizard adds a unique suffix to distinguish the business object. If an IDoc packet has two segments with the same name, for example segOrder, then 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 suffix appended to the name by the enterprise service discovery wizard.

Installing the adapter

The WebSphere Adapter for SAP Software has its own Installer, which places a resource adapter archive (RAR) and other adapter artifacts on your system. Before installing the adapter, you must have access to an IBM WebSphere Adapters, Version 6.0 product CD. The CD contains the setup launchpad and Installer.

For information about installing the adapter, see Installing IBM WebSphere Adapters

Adapter environment

Before installing, configuring, and using the adapter, you must understand its environment requirements.

Hardware and software requirements

For hardware and software requirements for this adapter, see IBM WebSphere Adapters and IBM WebSphere Business Integration Adapters: Hardware and Software Requirements. Select your adapter from the list of WebSphere adapters.

Adapter dependencies

- Add the SAP Java API (also known as SAP Java Connector or SAP JCo) to your project, as part of the application deployment process.
- (Windows® users) Install the msvcp71.dll and msvcr71.dll files in the Windows system path. See SAP Note 684106 on the SAP Service Marketplace Web site (www.service.sap.com) for additional details. The DLL files are contained in the attachments section of the SAP Note along with instructions on how to install them. You must have a valid user name and password for the SAP Service Marketplace to view the SAP Note.
- Set up a CPIC user account in the SAP system to be used by the adapter and the enterprise service discovery wizard to connect to SAP.
- If using the ALE interface, review the ALE interface prerequisites

Installed file structure

After you install the adapter, you can view the installed files and directories, all of which have the installation directory as their root. For example, if the installation directory for the adapter is c:\WebSphereBI, then the CWYAP_SAPAdapter.rar file has the following absolute path: c:\WebSphereBI\adapter\SAP\deploy\CWYAP_SAPAdapter.rar. For the same installation directory, the \adapter\SAP\deploy\CWYAP_SAPAdapter_Tx.rar file has the following absolute path: c:\WebSphereBI\adapter\SAP\deploy\CWYAP_SAPAdapter_Tx.rar.

Note the following:

- The adapter RAR file contains both the adapter and the enterprise service discovery wizard files.
- UNIX® and Windows platforms share the same installed directory and file structure, with the only difference being the directory path designation (forward slash / for UNIX, backslash \ for Windows).

Directory and file structure for UNIX/Linux

The following table lists the UNIX/Linux directories and files for the WebSphere Adapter for SAP Software. Directories and files are grouped into categories.

Installed files and directories (UNIX/Linux)

File and directory category	Directories and files
RAR files	/adapter/SAP/deploy/CWYAP_SAPAdapter.rar: Use this file if you want the adapter to perform ALE and BAPI processing without local transaction support. In this case, the application provides the local transaction support. /adapter/SAP/deploy/CWYAP_SAPAdapter_Tx.rar: Use this file if you want the container (WebSphere Process Server) to control local transaction support for BAPI processing. In this case, the adapter participates in the local transaction started by the container.
Sample files	/adapter/SAP/samples/CWYAP_EMDSample.zip /adapter/SAP/samples/CWYAP_NonEMDSample.zip
Notices file	/adapter/SAP/notices.txt
ISA plug-in zip file	/adapter/SAP/ISAPlugin/com.ibm.com.esupport.client.SS6FE6_RASAP.zip
IBM Tivoli® License Manager (ITLM) file	/adapter/SAP/5724L79E060000.sys
Log Message zip file	/adapter/SAP/messages/CWYBS_AdapterFoundation_messages.zip /adapter/SAP/messages/CWYBS_AdapterFoundation_messages.tar /adapter/SAP/messages/CWYAP_SAPAdapter_messages.zip /adapter/SAP/messages/CWYAP_SAPAdapter_messages.tar /adapter/SAP/messages/CWYAP_SAPAdapter_Tx_messages.zip /adapter/SAP/messages/CWYAP_SAPAdapter_Tx_messages.tar

Directory and file structure for Windows

The following table lists the Windows directories and files for the WebSphere Adapter for SAP Software. Directories and files are grouped into categories.

Installed files and directories (Windows)

File and directory category	Directories and files
RAR files	\adapter\SAP\deploy\CWYAP_SAPAdapter.rar: Use this file if you want the adapter to perform ALE and BAPI processing without local transaction support. In this case, the application provides the local transaction support. \adapter\SAP\deploy\CWYAP_SAPAdapter_Tx.rar: Use this file if you want the container (WebSphere Process Server) to control local transaction support for BAPI processing. In this case, the adapter participates in the local transaction started by the container.
Sample files	\adapter\SAP\samples\CWYAP_EMDSample.zip \adapter\SAP\samples\CWYAP_NonEMDSample.zip
Notices files	\adapter\SAP\notices.txt

File and directory category	Directories and files
ISA plug-in zip file	\adapter\SAP\ISAPlugin\com.ibm.com.esupport.client.SS6FE6_RASAP.zip
IBM Tivoli License Manager (ITLM) file	\adapter\SAP\5724L79E060000.sys
Log Message zip file	\adapter\SAP\messages\CWYBS_AdapterFoundation_messages.zip \adapter\SAP\messages\CWYBS_AdapterFoundation_messages.tar \adapter\SAP\messages\CWYAP_SAPAdapter_messages.zip \adapter\SAP\messages\CWYAP_SAPAdapter_messages.tar \adapter\SAP\messages\CWYAP_SAPAdapter_Tx_messages.zip \adapter\SAP\messages\CWYAP_SAPAdapter_Tx_messages.tar

Deploying the adapter

After you install the WebSphere Adapter for SAP Software, you must deploy it. Deployment consists of creating a project, adding external dependencies to that project, configuring the service, and then deploying the application to start on WebSphere Process Server.

The adapter is distributed as a Resource Adapter Archive (RAR) file.

1. You install the adapter into WebSphere Integration Developer by importing the RAR file.
2. Once the adapter has been installed, you generate an Enterprise Application Archive (EAR) file.
3. The EAR file is then deployed to WebSphere Process Server using the administrative console.

Note that, while WebSphere Integration Developer only runs on Windows or Linux®, WebSphere Process Server runs on Windows, Linux and UNIX platforms.

In principle, deploying the adapter is the same as deploying any other component on WebSphere Process Server. For more information on deploying components on WebSphere Process Server, see the WebSphere Integration Developer documentation.

Deployment prerequisites

You must install these products before you can deploy the adapter:

- WebSphere Integration Developer V6.0 (WebSphere Integration Developer)
- WebSphere Adapter for SAP Software, installed on the same machine as WebSphere Integration Developer
- WebSphere Process Server administrative console

For WebSphere Process Server installation instructions, see the WebSphere Process Server documentation.

In addition to installing these products, make sure you know the following information for accessing the SAP application:

- UserID
- SAP password
- SAP host name (or IP address)

- SAP system number (usually 00)
- SAP client number (usually 100)

Creating the project

The first step of deploying the adapter is to import the adapter .RAR file that was installed during installation into WebSphere Integration Developer and create the project in WebSphere Integration Developer.

The following steps are performed using WebSphere

Integration Developer. For details about this tool, refer to the WebSphere Integration Developer documentation.

1. Click **Start** → **Programs** → **IBM WebSphere** → **Integration Developer 6.0** to launch WebSphere Integration Developer.
2. Click to **Window** → **Open perspective** → **Other** → **J2EE** to switch to the J2EE perspective.
3. Right-click **Connector projects** and select **File** → **Import** from the pop-up menu.
4. Select the location from where you will import the .RAR file (the same location where you copied your adapter file during installation), and specify a project name.
5. Deselect the **Add module to an EAR project** check box.
6. Click **Finish** to import the RAR file. This creates a new J2EE Connector project in the workspace.
7. Shut down and restart WebSphere Integration Developer.

The next step is to add external dependencies to the project.

Adding external dependencies

After you create the adapter application project, you must add the required external dependencies into the project. SAP Java Connector (SAP JCo) interface is an external dependency that the adapter has for connecting to the SAP software application. The adapter uses this interface to make calls to the SAP native interfaces.

Use WebSphere Integration Developer to add the SAP Java Connector (SAP JCo) interface to the imported project. All external libraries and JAR files must first be copied to the appropriate locations on WebSphere Process Server:

- Copy the dependencies libraries (*.dll, *.so, and *.o files) to the <*WPS_INSTALL*>\bin directory.
- (Windows users) Install the msvcp71.dll and msvcr71.dll files in the Windows system path. See SAP Note 684106 on the SAP Service Marketplace Web site (www.service.sap.com) for additional details. The DLL files are contained in the attachments section of the SAP Note along with instructions on how to install them. You must have a valid user name and password for the SAP Service Marketplace to view the SAP Note.
- Copy sapjco.jar to the <*WPS_INSTALL*>\lib directory.

Where <*WPS_INSTALL*> represents the WebSphere Process Server installation directory.

Follow these steps to add the sapjco.jar file to the project.

1. In the J2EE perspective of WebSphere Integration Developer, click **Connector Properties**.
2. Right-click **CWYAP_SAPAdapter** and then click **Properties**. The Java Build Path window appears.
3. In the left-side pane of the Java Build Path window, click **Java Build Path**.
4. In the right-side pane, click the **Libraries** tab and then click **Add External Jars**.
5. Choose the file sapjco.jar and click **Open**. The file sapjco.jar should now appear in the list of **JARs and class folders in the build path**.
6. Click **OK**. The sapjco.jar file is now part of your connector project and appears in the Project Explorer window of WebSphere Integration Developer.

The next step is to configure the service, a process that includes configuring the adapter and creating the business objects that the adapter will exchange with the SAP software application.

Configuring the service

The configuration process is done using the enterprise service discovery wizard in WebSphere Integration Developer. As you complete the process you will enter all the information necessary to configure the adapter for the first time. The output from the enterprise service discovery wizard is saved to a business integration module, which contains the business objects, the import file (which describes outbound processing, as defined by the ManagedConnectionFactory specification), the export file (which describes inbound event processing, as defined by the ActivationSpec), and the Web Services Description Language (WSDL) file.

Configuring business objects

When you deploy the adapter, you may choose to create objects for the following SAP components:

- Simple BAPI
- BAPI transaction
- Inbound ALE IDoc
- Outbound ALE IDoc
- Outbound ALE packet

Configuring adapter properties

Note the following issues regarding configuring adapter properties:

- **J2C activation specification properties**: During deployment, if you specify J2C activation specification (ActivationSpec) properties when you initially configure the service, those property settings will remain in place (in other words, you cannot update the properties later, after you install the application, via the WebSphere Process Server administrative console). If, for any reason, you want to set the J2C activation specification properties after installing the application via the administrative console, then do not set them during deployment. Note that J2C connection factory properties *can* be set during deployment and then updated via the administrative console after you deploy the application. Refer to the list of J2C activation specification properties for a complete listing of the properties that cannot be reconfigured after you install the application.
- **J2C connection factory properties**: The following J2C connection factory properties are not configured during deployment using the enterprise service

discovery wizard, but rather are configured later, after deployment, using the WebSphere Process Server administrative console:

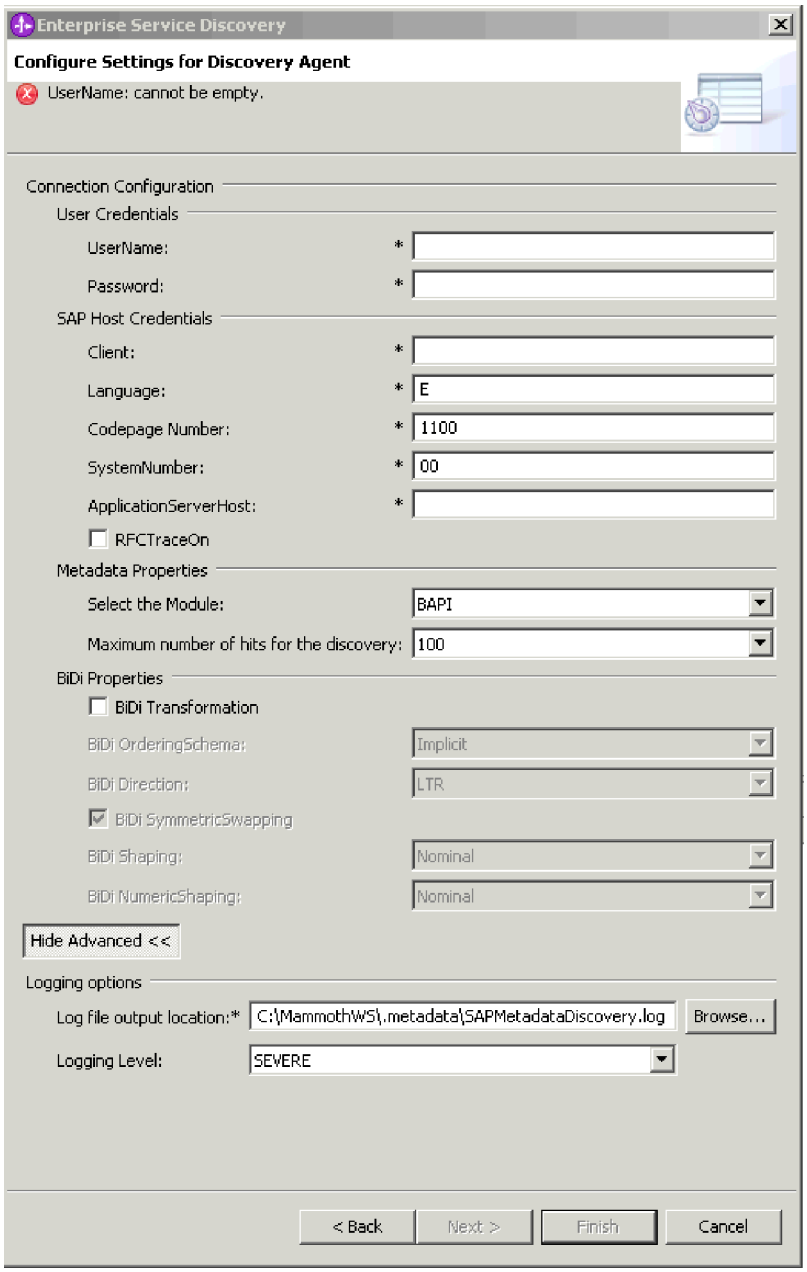
- GatewayHost
- GatewayService
- Group
- MessageServerHost
- RFCTraceOn
- SAPSystemID

Configuring the service for a simple BAPI

The configuration process described in the following steps configures the adapter and creates business objects for different BAPI calls that are associated with a customer.

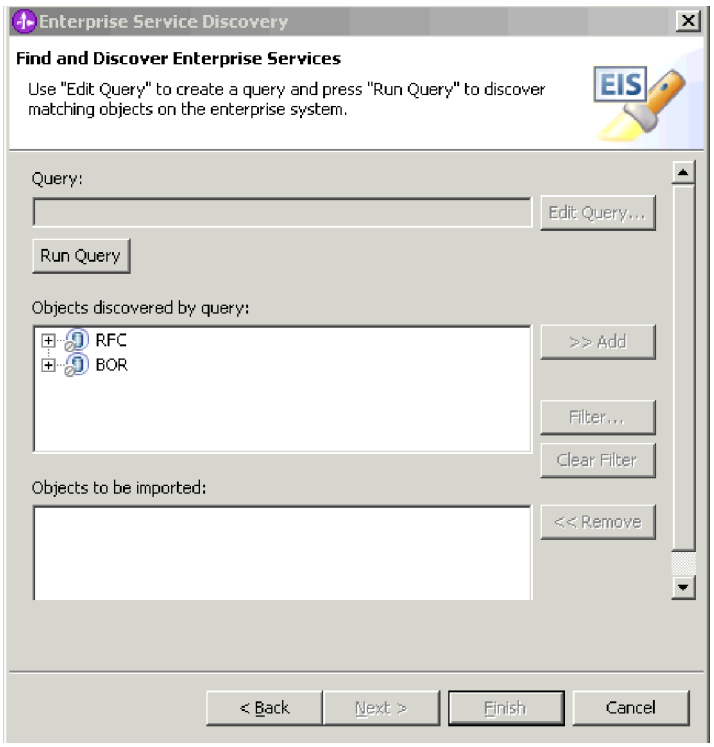
For information about configuring other business objects, refer to the appropriate service configuration topics.

1. Switch to the Business Integration perspective in WebSphere Integration Developer.
2. Right-click the frame of the Business Integration perspective window and select **New → Enterprise Service Discovery** from the pop-up menu. If **Enterprise Service Discovery** is not visible, select **Other** from the bottom of the pop-up menu. Then, in the window that appears, expand the Business Integration folder and select **Enterprise Service Discovery** and click **Next**.
3. When prompted to select an adapter to use for discovering the service, select **IBM WebSphere Adapter for SAP Software** and click **Next**. If you have previously run the enterprise service discovery wizard, your connection properties have been saved and will appear when you expand the adapter name node by clicking the plus symbol (+) next to the adapter name. You can select the saved connection properties if you plan to connect to the same SAP application as when you last ran the enterprise service discovery wizard.
4. When prompted to specify properties in the Configure Settings for Discovery Agent window, specify the connection configuration properties for connecting to SAP. Be sure to set the **Select Module** property (under **Metadata Properties**) to the value of `BAPI`. Properties marked with an asterisk (*) are mandatory.



Specifying properties

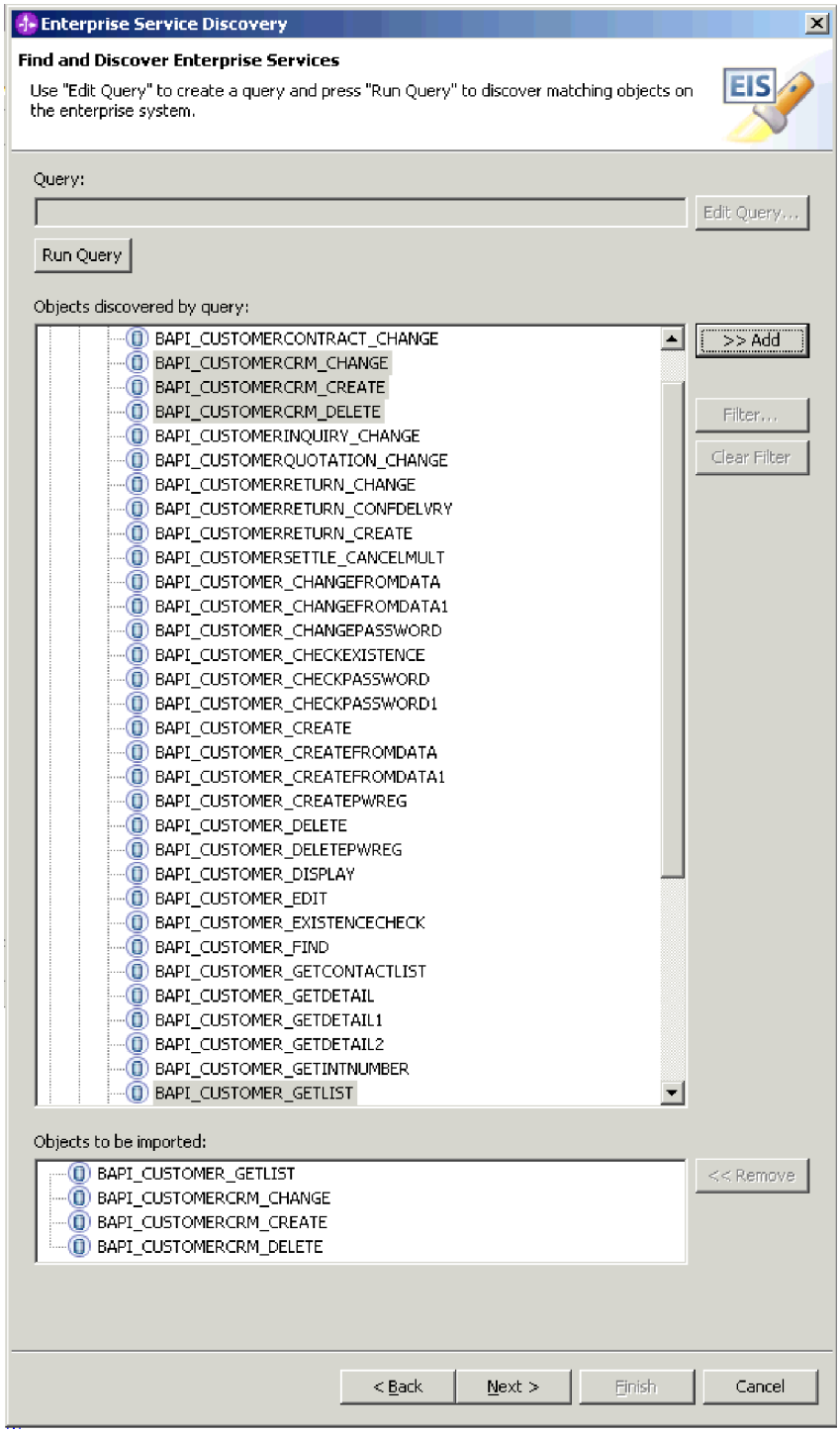
5. At the bottom of the window, click the **Show Advanced** button.
6. When prompted to specify logging options, specify a log file location and set the **Logging Level**. In a test environment, choose `FINEST`, which provides the highest level of logging. In a production environment, choose a level lower than `FINEST`, so as to optimize the logging process.
7. Click **Next**.
8. In the Find and Discover Enterprise Services window, click **Run Query**. The objects discovered by the query are grouped into two BAPI categories: RFC and BOR.



BAPI categories

9. Under **Objects Discovered by query**, select RFC or BOR, drill down to the **Discover By Name** node, and click the **Filter** button. Or, you can drill down to **Discover By Description**.
10. In the Filter Properties for Discover by Name window, enter `BAPI_CUSTOMER*`, which is the name of the BAPI in SAP plus an asterisk as a wild card character to indicate that you want to discover all SAP application components that start with BAPI_CUSTOMER.
11. Click **OK**.
12. Select the following BAPIs:
    - BAPI_CUSTOMERCRM_CREATE
    - BAPI_CUSTOMERCRM_CHANGE
    - BAPI_CUSTOMERCRM_DELETE
    - BAPI_CUSTOMER_GETLIST
13. In the Configuration Parameters window, do the following to add the selected BAPIs to the list of business objects to be imported:
    a. Select the **Use Field Name to generate attribute(s)** check box.
    b. Select the **Check if you want to select optional parameters for this interface** check box and then select the optional parameters you want included in your business object definition. By default, enterprise service discovery generates the mandatory parameters for the selected BAPI interface, so select this check box to also include the optional parameters.
    c. Click **OK**.

    The selected objects appear in the bottom window frame.

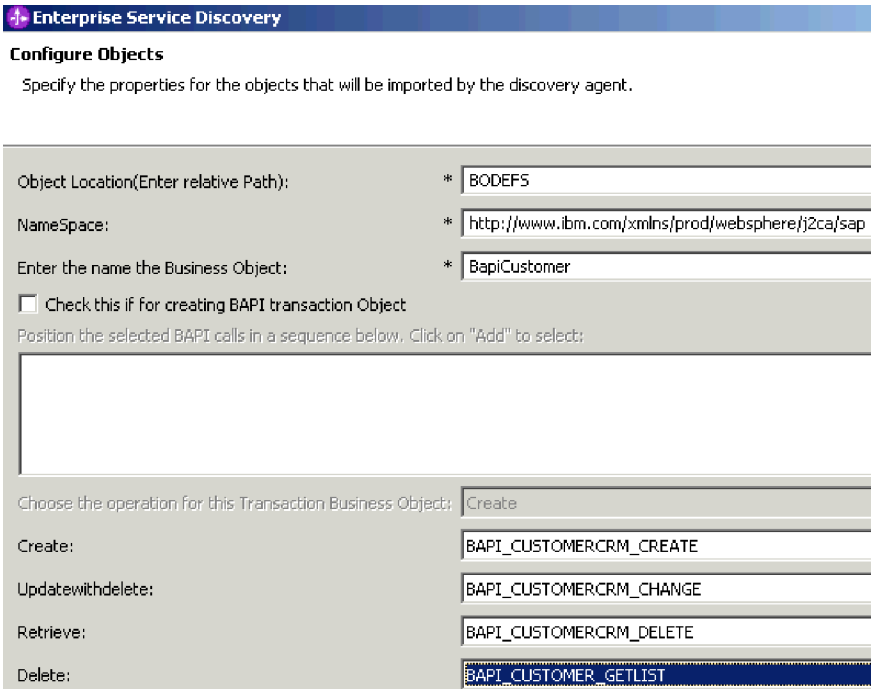


Business objects to be imported

14. Repeat step 12 and step 13 for the following objects and then click **Next**:
    - BAPI_CUSTOMERCRM_CREATE
    - BAPI_CUSTOMERCRM_CHANGE
    - BAPI_CUSTOMERCRM_DELETE
    - BAPI_CUSTOMER_GETLIST

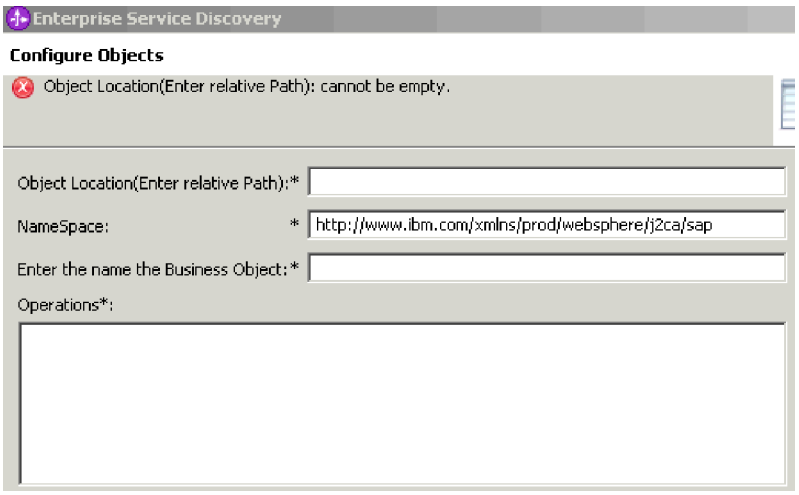
    The selected objects appear in the bottom window frame.

15. If you want to remove an object from the list, select the object name and click **Remove**.
16. After you have added all the objects to be discovered, click **Next** to proceed.
17. In the Configure Objects window:
    a. Enter the directory name, for example `BODEFS`, for the object location. This specifies a directory that is relative to the project's directory in the WebSphere Integration Developer workspace.
    b. Specify the name space.
    c. Enter the name of the business object. For example, BapiCustomer.
    d. Do *not* select the **Check this for creating BAPI transaction object** check box.



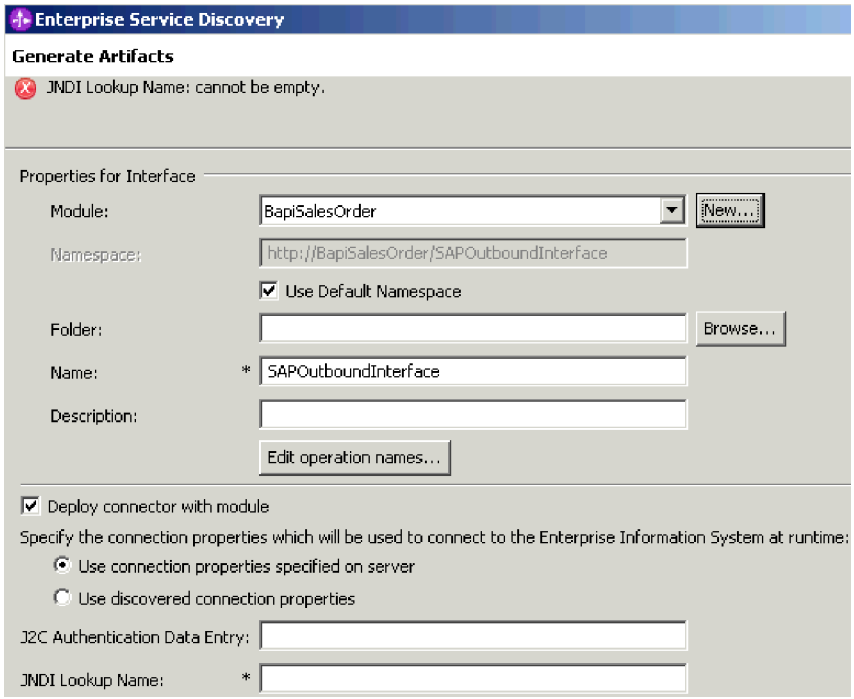
Configure Objects window for multiple BAPI selection

If in Step 12 you selected a single BAPI, for example BAPI_CUSTOMERCRM_CREATE, then instead of the previous screen, the following screen appears. Use the following screen to specify the object location and business object name and click **Add** to add the operation(s) you want to associate to the single BAPI.



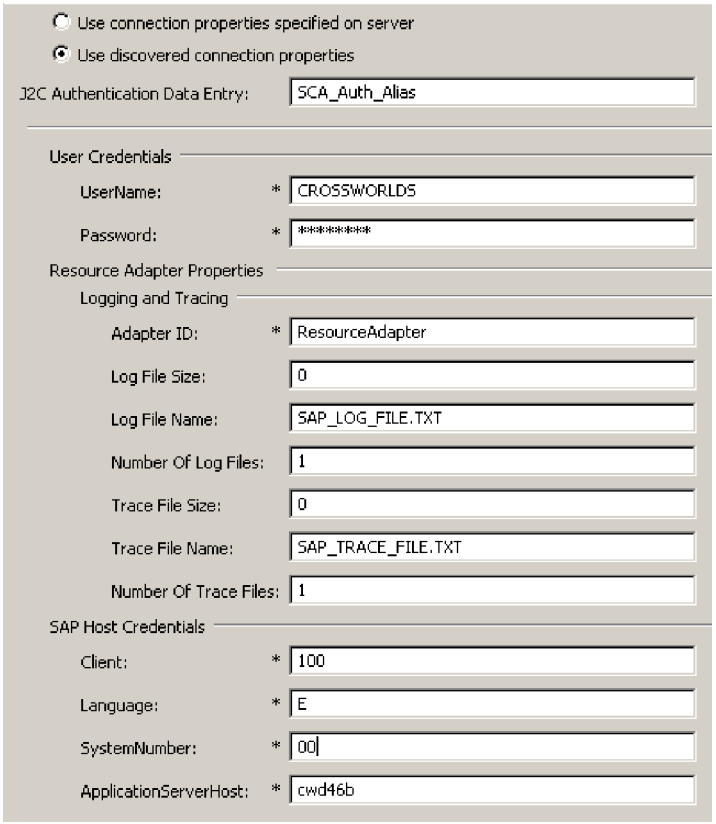
Configure Objects window for single BAPI selection

18. Specify the appropriate JCo methods for each of the object's operations. For example, specify the following methods:
    - For the CREATE operation, specify `BAPI_CUSTOMERCRM_CREATE`.
    - For the UPDATEWITHDELETE operation, specify `BAPI_CUSTOMERCRM_CHANGE`.
    - For the RETRIEVE operation, specify `BAPI_CUSTOMER_GETLIST`.
    - For the DELETE operation, specify `BAPI_CUSTOMERCRM_DELETE`.
19. Click **Next**.
20. In the Generate Artifacts window, click **New** to create a new business integration module and then specify `BapiCustomer` for the module name where the SCA artifacts (business objects, their properties, import file, export file, and WSDL) should be saved.



Generate Artifacts window

21. In the Generate Artifacts window, specify the folder within the module where the service description should be saved.
22. In the **J2C Authentication Data Entry** field, enter `SAP_Auth_Alias` and select the **Deploy connector with module** check box.
23. Click the **Use discovered connection properties** radio button to set properties at this time. (The **Use connection properties specified on server** button lets you configure properties later, using the WebSphere Process Server administrative console).



Specifying properties

24. Specify the connection properties and then click **Finish**. Properties marked with an asterisk (*) are required.

The new BapiCustomer module is added to the Business Integration perspective, along with all its artifacts.

After you configure the service, the next step is to generate reference bindings. The purpose of reference bindings is to link the adapter to other server processes.

Configuring the service for a BAPI transaction

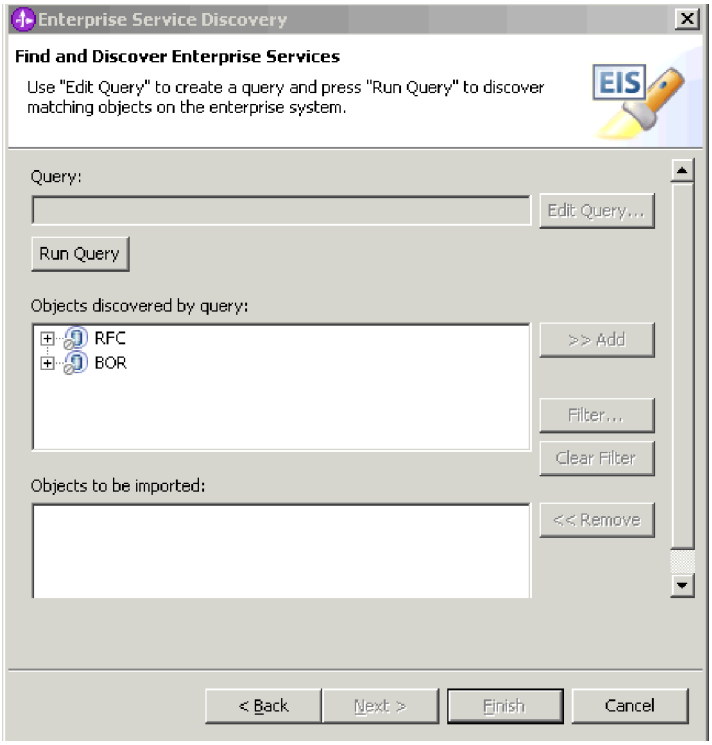
The configuration process described in the following steps configures the adapter and creates business objects for a BAPI transaction.

For information about configuring other business objects, refer to the appropriate service configuration topics.

1. Switch to the Business Integration perspective in WebSphere Integration Developer.
2. Right-click the frame of the Business Integration perspective window and select **New → Enterprise Service Discovery** from the pop-up menu. If

Enterprise Service Discovery is not visible, select **Other** from the bottom of the pop-up menu. Then, in the window that appears, expand the Business Integration folder and select **Enterprise Service Discovery** and click **Next**.

3. When prompted to select an adapter to use for discovering the service, select **IBM WebSphere Adapter for SAP Software** and click **Next**. If you have previously run the enterprise service discovery wizard, your connection properties have been saved and appear when you expand the adapter name node by clicking the plus symbol (+) next to the adapter name. You can select the saved connection properties if you plan to connect to the same SAP application as when you last ran the enterprise service discovery wizard,
4. When prompted to specify properties in the Configure Settings for Discovery Agent window, specify the adapter configuration properties for connecting to SAP. Properties marked with an asterisk (*) are mandatory. Note that when you first created the project, if you imported a RAR file that supports transactions, the **Select Module** property (under **Metadata Properties**) is set to a value of `BAPI` and cannot be changed. Otherwise, set the value of this property to `BAPI`.
5. At the bottom of the window, click the **Show Advanced** button.
6. When prompted to specify logging options, specify a log file location and set the **Logging Level**. In a test environment, choose `FINEST`, which provides the highest level of logging. In a production environment, choose a level lower than `FINEST`, so as to optimize the logging process.
7. Click **Next**.
8. In the Find and Discover Enterprise Services window, click **Run Query**. The objects discovered by the query are grouped into two BAPI categories: RFC and BOR.

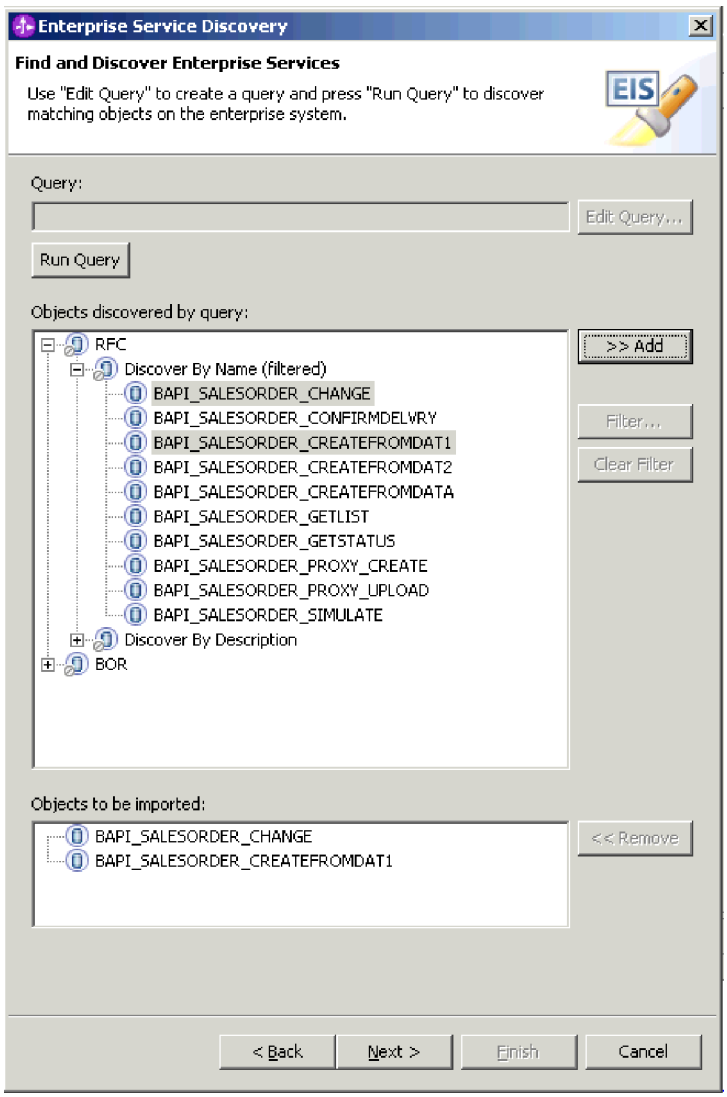


BAPI categories

9. Under **Objects Discovered by query**, select RFC or BOR, drill down to the **Discover By Name** node, and click the **Filter** button. Or, you can drill down to **Discover By Description**.

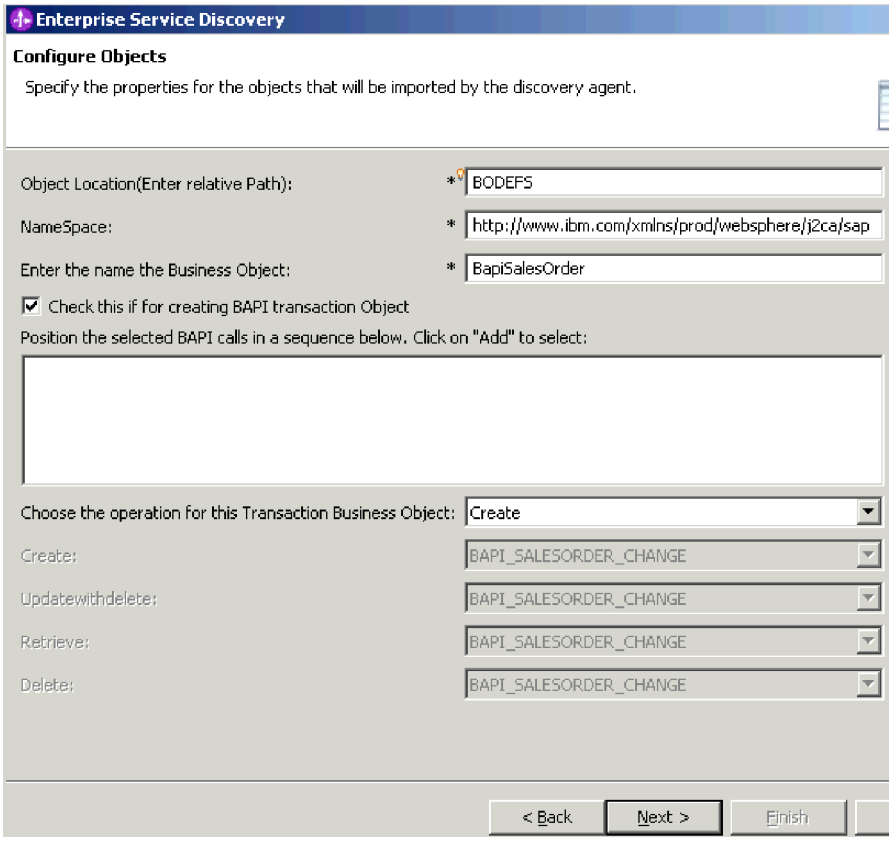
10. In the Filter Properties for Discover by Name window, enter the name of the BAPI you want to add to your transaction, plus an asterisk as a wild card character to indicate that you want to discover all SAP application components that start with BAPI_SALESORDER. For example, enter `BAPI_SALESORDER*`
11. Click **OK**.
12. Select the BAPIs to add to the transaction. For example, select BAPI_SALESORDER_CREATE and BAPI_SALESORDER_CHANGE.
13. Repeat step 12 for the remaining BAPIs you want to add to your transaction, and then click **Next**.
14. In the Configuration Parameters window, do the following to add the selected BAPIs to the list of business objects to be imported:
    a. Select the **Use Field Name to generate attribute(s)** check box.
    b. Select the **Check if you want to select optional parameters for this interface** check box and then select the optional parameters you want included in your business object definition. By default, enterprise service discovery generates the mandatory parameters for the selected BAPI interface, so select this check box to also include the optional parameters.
    c. Click **OK**.

    The selected objects appear in the bottom window frame.



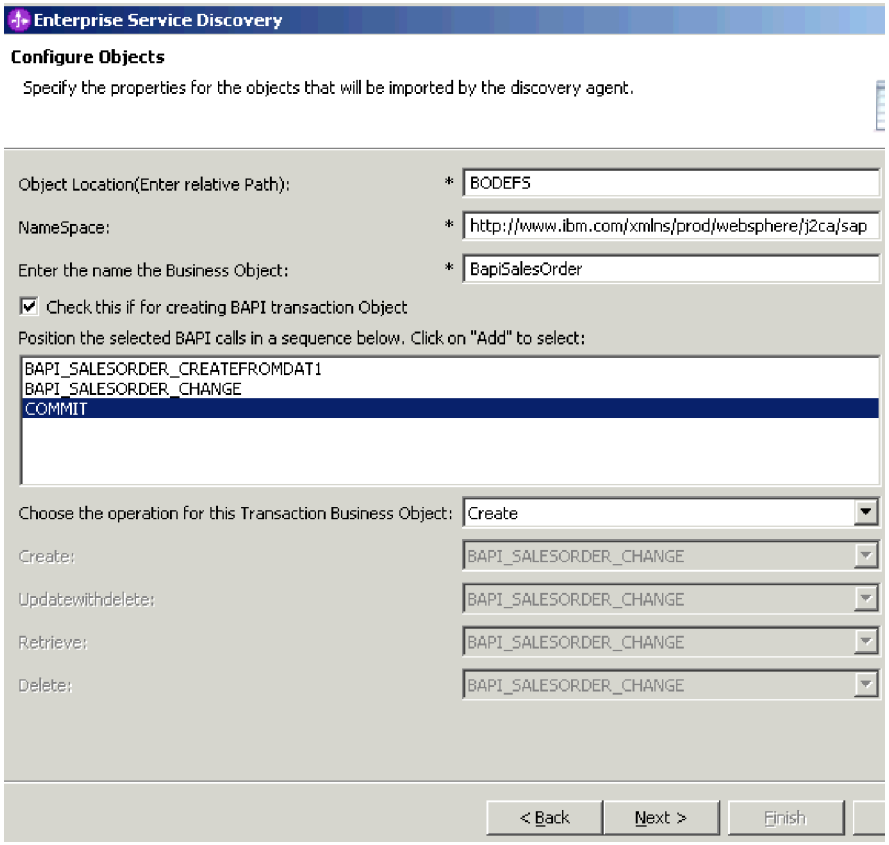
Selected BAPI objects

15. If you want to remove an object from the list, select the object name and click **Remove**.
16. Click **Next** to proceed.
17. In the Configure Objects window:



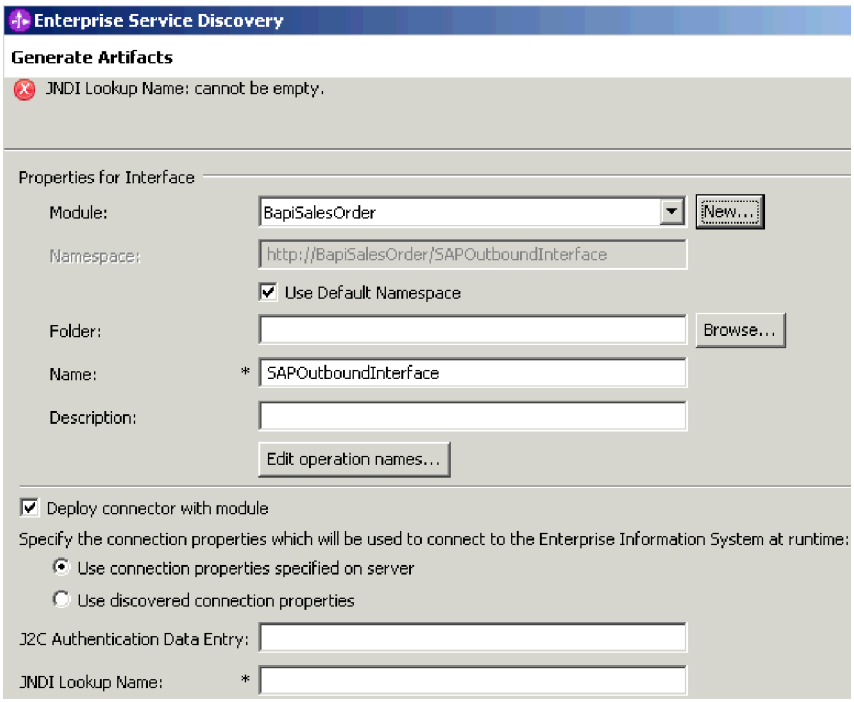
a. Enter the directory name, for example `BODEFS`, for the object location. This specifies a directory that is relative to the project's directory in the WebSphere Integration Developer workspace.
b. Specify the name space.
c. Enter the name of the business object.
d. Select the **Check this for creating BAPI transaction object** check box.

18. Click **Add**.
19. Perform the following steps to specify the sequence of BAPIs in the transaction object:
    a. Select a BAPI.
    b. Click **Add**.
    c. Select `COMMIT` as needed for the transaction.
    d. Repeat these steps for each BAPI you want to include in the transaction. The list of BAPIs you create must be in the sequence in which they are to be performed within the transaction.



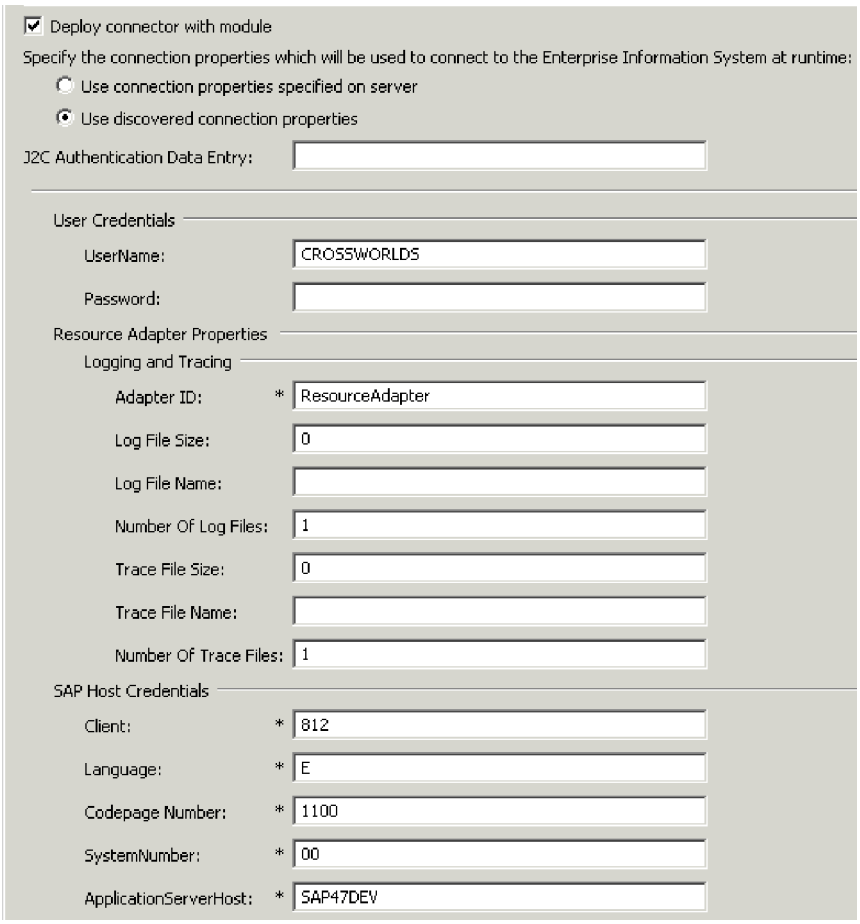
Configure objects window

20. Click **Next**.
21. In the Generate Artifacts window, click **New** to create a new business integration module and then specify the module name (for example, `BapiSalesOrder`) where the SCA artifacts (business objects, their properties, import file, export file, and WSDL) should be saved.



Generate Artifacts window

22. In the Generate Artifacts window, specify the folder within the module where the service description should be saved.
23. In the **J2C Authentication Data Entry** field, enter `SAP_Auth_Alias` and select the **Deploy connector with module** check box.
24. In the Generate Artifacts window, click the **Use discovered connection properties** radio button to set properties at this time. (The **Use connection properties specified on server** button lets you configure properties later, using the WebSphere Process Server administrative console).



Generate Artifacts window

25. Specify the connection properties and then click **Finish**. Properties marked with an asterisk (*) are required.

The new BapiCustomer module is added to the Business Integration perspective, along with all its artifacts.

After you configure the service, the next step is to generate reference bindings. The purpose of reference bindings is to link the adapter to other server processes.

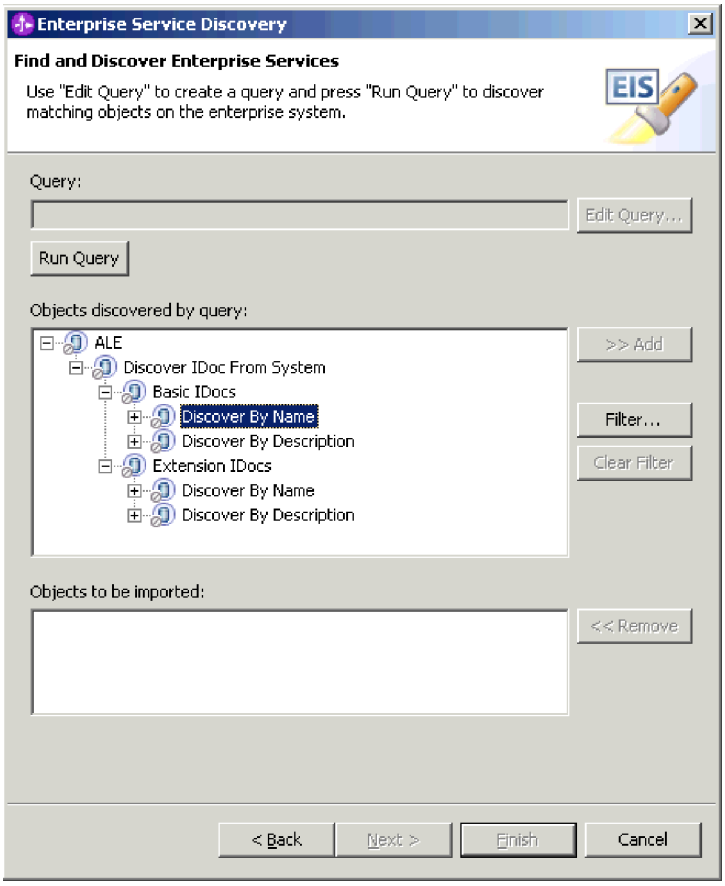
Configuring the service for an outbound ALE IDoc and IDoc packet

The configuration process described in the following steps configures the adapter, creates business objects for outbound ALE IDocs, and if desired, creates an IDoc wrapper business object for an IDoc packet of multiple individual IDocs.

For information about configuring other business objects, refer to the appropriate service configuration topics.

1. Switch to the Business Integration perspective in WebSphere Integration Developer.
2. Right-click the frame of the Business Integration perspective window and select **New → Enterprise Service Discovery** from the pop-up menu. If **Enterprise Service Discovery** is not visible, select **Other** from the bottom of the pop-up menu. Then, in the window that appears, expand the Business Integration folder and select **Enterprise Service Discovery** and click **Next**.

3. When prompted to select an adapter to use for discovering the service, select **IBM WebSphere Adapter for SAP Software** and click **Next**. If you have previously run the enterprise service discovery wizard, your connection properties have been saved and appear when you expand the adapter name node (click the plus symbol + next to the adapter name). You can select the saved connection properties if you plan to connect to the same SAP application as when you last ran the enterprise service discovery wizard.
4. When prompted to specify properties in the Configure Settings for Discovery Agent window, specify the adapter configuration properties for connecting to SAP. Properties marked with an asterisk (*) are mandatory. Be sure to set the **Select Module** property (under **Metadata Properties**) to the value of `ALE`.
5. At the bottom of the window, click the **Show Advanced** button.
6. When prompted to specify logging options, specify a log file location and set the **Logging Level**. In a test environment, choose `FINEST`, which provides the highest level of logging. In a production environment, choose a level lower than `FINEST`, so as to optimize the logging process.
7. Click **Next**.
8. In the Find and Discover Enterprise Services window, click **Run Query**.
9. Under **Objects Discovered by query**, drill down to the **Discover By Name** node, and then click the **Filter** button. You can also drill down to **Discover By Description**, as illustrated in the following screen.

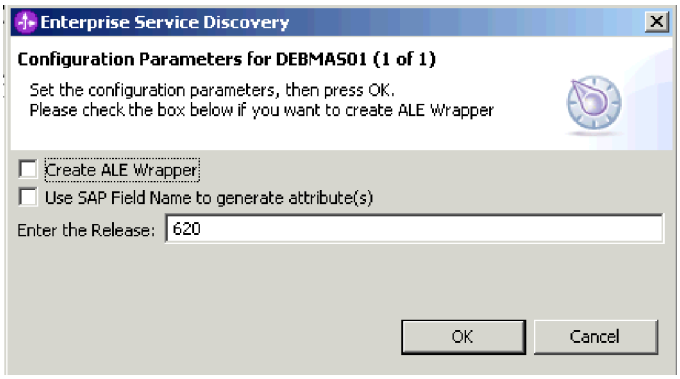


Drilling down to discover by description

10. In the Filter Properties for Discover by Name window, enter the name of the IDoc you want to discover. You can include an asterisk (*) as a wild card

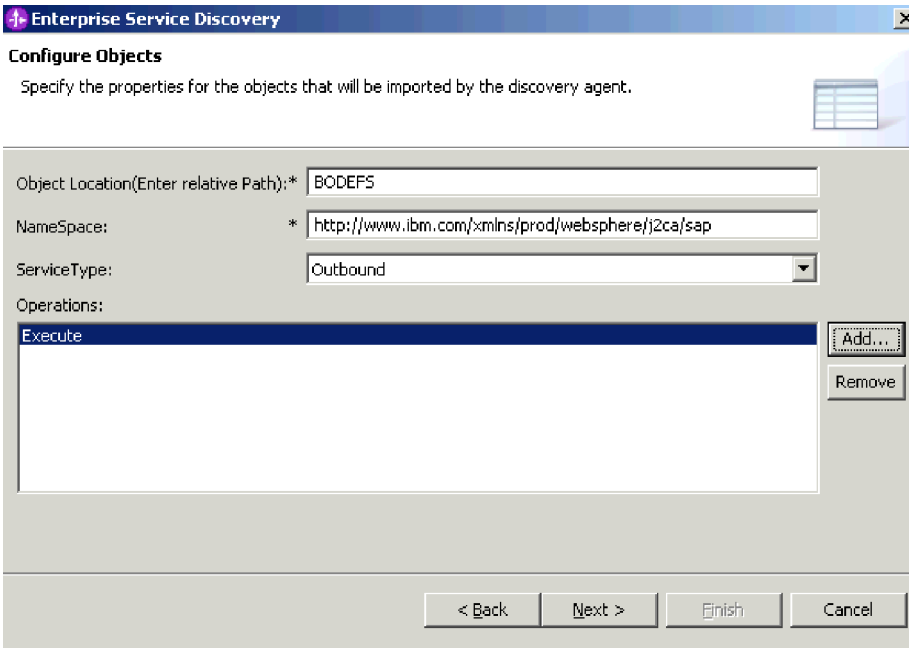
character at the end of the name to indicate that you want to discover all SAP application components that start with the specified name.

11. Click **OK**.
12. Navigate to the desired Basic or Extension IDoc and click the **Add** button. The Configuration Parameters window appears.



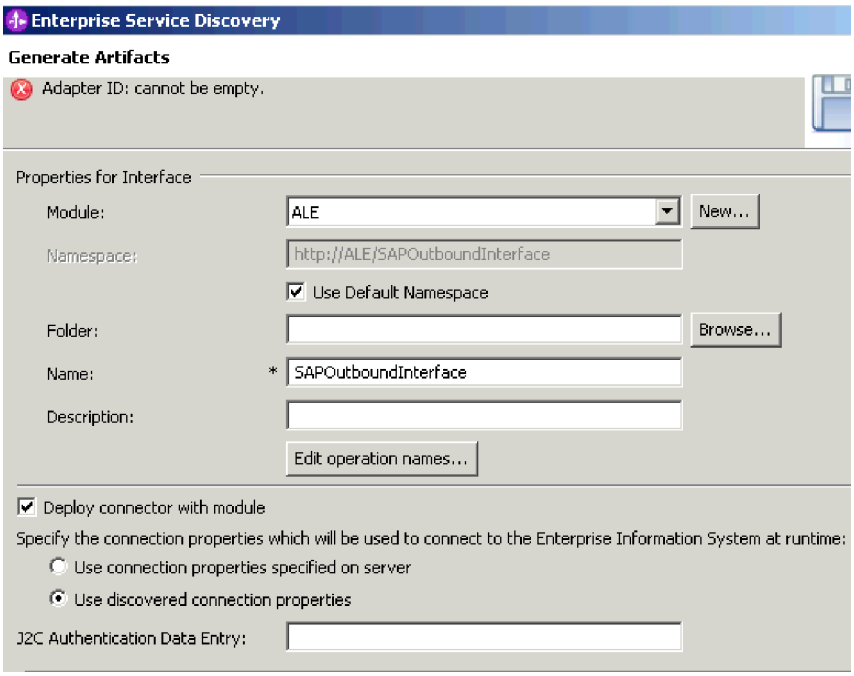
Configuration Parameters window

13. In the Configuration Parameters window, do the following to add the IDoc to the list of business objects to be imported.
    a. If you want to create an ALE wrapper object (for an IDoc packet of multiple individual IDocs), select the **Create ALE wrapper** check box.
    b. Select the **Use SAP Field Name to generate attribute(s)** check box.
    c. In the **Enter the Release** field, specify the SAP release number to identify the IDoc type you want the enterprise service discovery wizard to use for creating business objects. Note that you can specify an earlier release than the one you are currently using, if for some reason you wish to create business objects based on earlier versions of the IDoc type. If the earlier version of the IDoc type has fewer segments than the current version, the enterprise service discovery wizard may create a definition with missing segments or it may display an error indicating that the generation of the business object definition was unsuccessful. This inconsistency is due to different versions of SAP requiring different API calls.
    d. Click **OK**.
14. Repeat step 12 and step 13 for each IDoc you want to discover and then click **Next**. The selected objects appear in the bottom window frame.
15. If you want to remove an object from the list, select the object name and click **Remove**.
16. Click **Next** to proceed.
17. In the Configure Objects window, enter `BODEFS` in the Object Location field, specify the name space, and select `Outbound` in the **ServiceType** field.
18. Click **Add**. The Add window appears with a list of operations you can select to associate with this business object. For outbound, the only operation you can select is `Execute`.
19. Select the operation and click **OK**. The Configure Objects window appears with the selected operation listed under **Operations**.



Configure Objects window

20. Click **Next**.
21. In the Generate Artifacts window, click **New** and then specify the module name where the SCA artifacts (business objects, their properties, import file, export file, and WSDL) should be saved.



Generate Artifacts window

22. In the Generate Artifacts window, specify the folder within the module where the service description should be saved.
23. In the **J2C Authentication Data Entry** field, enter `SAP_Auth_Alias` and select the **Deploy connector with module** check box.

24. In the Generate Artifacts window, click the **Use discovered connection properties** radio button to set properties at this time. (The **Use connection properties specified on server** button lets you configure properties later, using the WebSphere Process Server administrative console).
25. Specify the connection properties and then click **Finish**. Properties marked with an asterisk (*) are required.

The new module is added to the Business Integration perspective, along with all its artifacts.

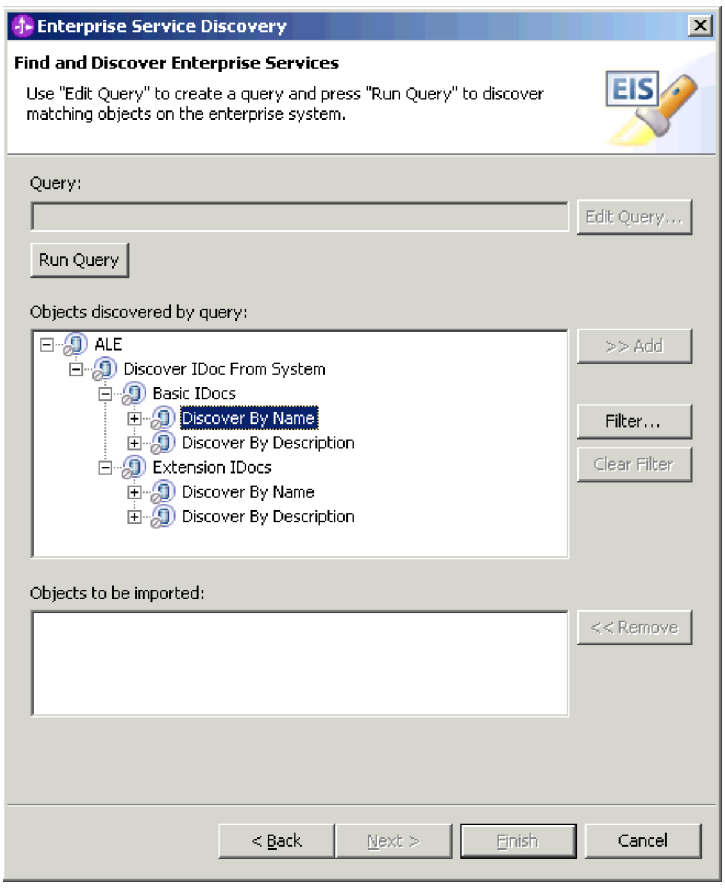
After you configure the service, the next step is to generate reference bindings. The purpose of reference bindings is to link the adapter to other server processes.

Configuring the service for an inbound ALE IDoc

The configuration process described in the following steps configures the adapter and creates business objects for inbound ALE IDocs.

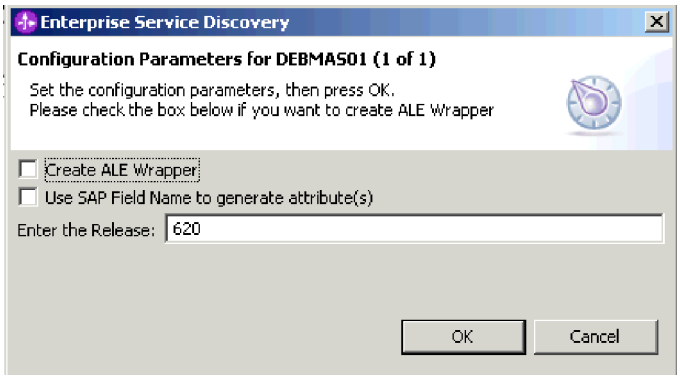
For information about configuring other business objects, refer to the appropriate service configuration topics.

1. Switch to the Business Integration perspective in WebSphere Integration Developer.
2. Right-click the frame of the Business Integration perspective window and select **New → Enterprise Service Discovery** from the pop-up menu. If **Enterprise Service Discovery** is not visible, select **Other** from the bottom of the pop-up menu. Then, in the window that appears, expand the Business Integration folder and select **Enterprise Service Discovery** and click **Next**.
3. When prompted to select an adapter to use for discovering the service, select **IBM WebSphere Adapter for SAP Software** and click **Next**. If you have previously run the enterprise service discovery wizard, your connection properties have been saved and appear when you expand the adapter name node (click the plus symbol + next to the adapter name). You can select the saved connection properties if you plan to connect to the same SAP application as when you last ran the enterprise service discovery wizard.
4. When prompted to specify properties in the Configure Settings for Discovery Agent window, specify the adapter configuration properties for connecting to SAP. Properties marked with an asterisk (*) are mandatory. Be sure to set the **Select Module** property (under **Metadata Properties**) to the value of `ALE`.
5. At the bottom of the window, click the **Show Advanced** button.
6. When prompted to specify logging options, specify a log file location and set the **Logging Level**. In a test environment, chose `FINEST`, which provides the highest level of logging. In a production environment, choose a level lower than `FINEST`, so as to optimize the logging process.
7. Click **Next**.
8. In the Find and Discover Enterprise Services window, click **Run Query**.
9. Under **Objects Discovered by query**, drill down to the **Discover By Name** node, and then click the **Filter** button. You can also drill down to **Discover By Description**, as illustrated in the following screen.



Discovering IDocs by name

10. In the Filter Properties for Discover by Name window, enter the name of the IDoc you want to discover. You can include an asterisk (*) as a wild card character at the beginning or end of the name to indicate that you want to discover all SAP application components that contain the specified name.
11. Click **OK**.
12. Navigate to the desired Basic or Extension IDoc and click the **Add** button. The Configuration Parameters window appears.



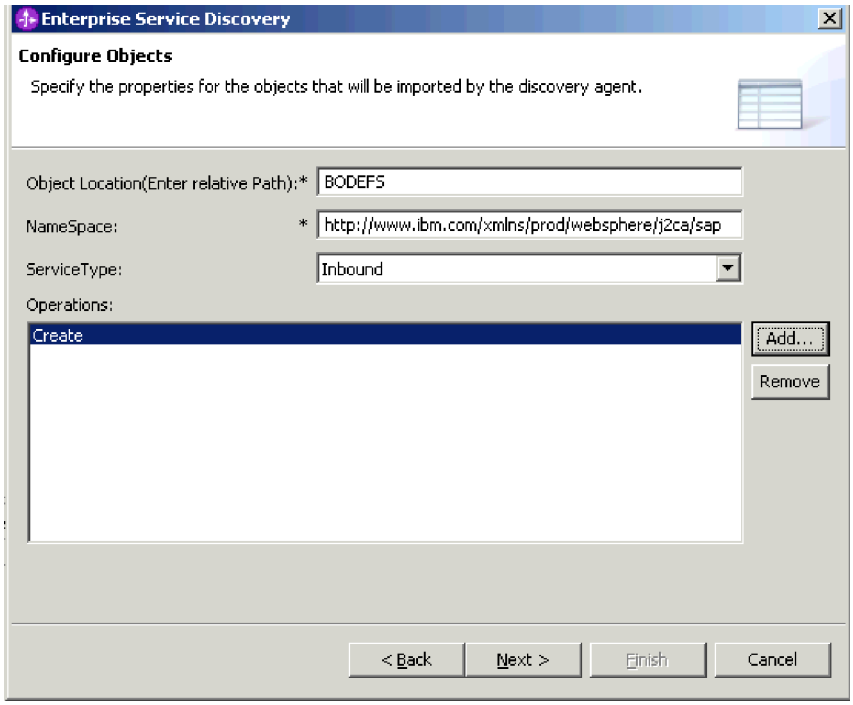
Configuration Parameters window

13. In the Configuration Parameters window, do the following to add the IDoc to the list of business objects to be imported.
    a. Do *not* select the **Create ALE wrapper** check box.
    b. Select the **Use SAP Field Name to generate attribute(s)** check box.

c. Do not change the value of the **Enter the Release** field.

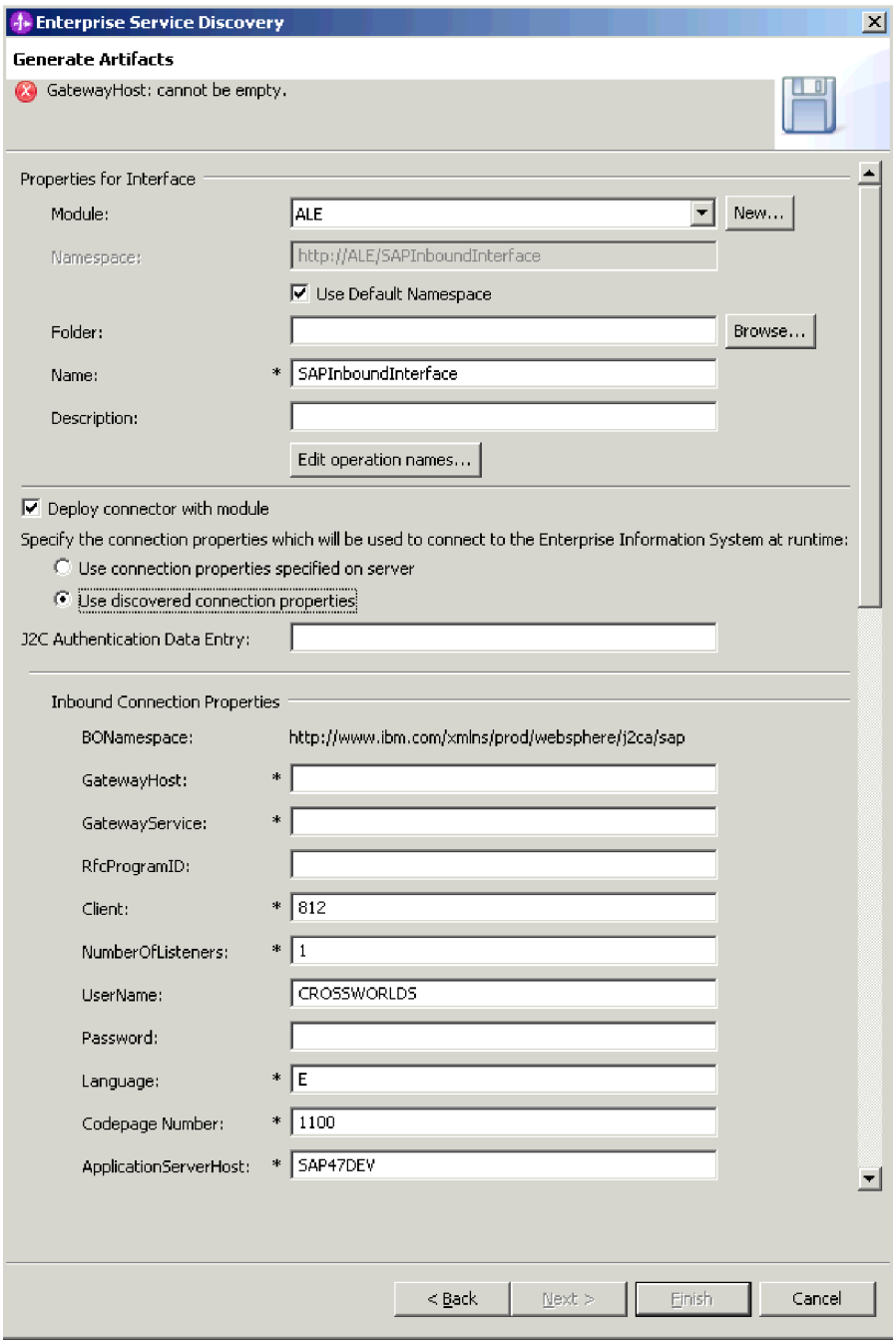
d. Click **OK**.

14. Repeat step 12 and step 13 for each IDoc you want to discover and then click **Next**. The selected objects appear in the bottom window frame.
15. If you want to remove an object from the list, select the object name and click **Remove**.
16. Click **Next** to proceed.
17. In the Configure Objects window, enter `BODEFS` in the Object Location field, specify the name space, and select `Inbound` in the **ServiceType** field.
18. Click **Add**. The Add window appears with a list of operations you can select to associate with this business object.
19. Select the operation and click **OK**. The Configure Objects window appears with the selected operation listed under **Operations**.
20. Select the operation and click **OK**. The Configure Objects window appears with the selected operation(s) listed under **Operations**.



Configure Objects window

21. Click **Next**.
22. In the Generate Artifacts window, specify the module name where the SCA artifacts (business objects, their properties, import file, export file, and WSDL) should be saved.



Generate Artifacts window

23. In the Generate Artifacts window, specify the folder within the module where the service description should be saved.
24. In the **J2C Authentication Data Entry** field, enter `SAP_Auth_Alias` and select the **Deploy connector with module** check box.
25. In the Generate Artifacts window, click the **Use discovered connection properties** radio button to set properties at this time. (The **Use connection properties specified on server** button lets you configure properties later, using the WebSphere Process Server administrative console).
26. Specify the connection properties and then click **Finish**. Properties marked with an asterisk (*) are required.

The new module is added to the Business Integration perspective, along with all its artifacts.

After you configure the service, the next step is to generate reference bindings. The purpose of reference bindings is to link the adapter to other server processes.

Generating reference bindings

Reference bindings are used by external WebSphere Business Integration SCA components to access the adapter. You create a reference to the adapter from the project module so as to link the adapter to the other server processes. This is required in a standalone testing environment only. It is not necessary when deploying the adapter in a production environment.

1. In the Business Integration Perspective of WebSphere Integration Developer, right-click the BapiCustomer module, and select **Open With → Assembly Editor**. The Assembly Diagram window appears with the module's Import component in view.
2. To create a new component, click the top-most icon in the left-side (vertical) pane of the Assembly Diagram Window. A new menu of icons appears.
3. Move the mouse pointer over each icon to display the hover Help and locate the icon that reads **Standalone References**.
4. Click the icon for **Standalone References**.
5. Click the blank area (right-side pane) of the Assembly Diagram window to drop the new **Standalone References** component into that pane.
6. Click the new **Standalone References** component. A yellow bulb displays on the right side of the component.
7. Drag and drop the yellow bulb around the new component to the import module. This draws a wire from the Import component to the new component and displays the Add Wire window.
8. In the Add Wire window, click **OK**. The new Standalone Reference component displays in the Assembly Diagram window with a "wire" that connects it to the module's Import component.
9. When prompted to use Java interfaces, click **No**.
10. Click **File → Save** to save the assembly diagram.

The WebSphere Business Integration project module is now created in build-time. Before you can start the application, you must export the project to an Enterprise Application Archive (EAR) file using WebSphere Integration Developer, and then install the EAR file using WebSphere Process Server administrative console.

Exporting the application

Before you can run the application, you must export the project to an EAR file using WebSphere Integration Developer.

1. In the J2EE perspective window of WebSphere Integration Developer, right-click on the application you want to export and select **Export** from the pop-up menu. The Export-Select window appears.
2. Select **EAR file** from the Export - Select window. The EAR Export window appears.
3. In the EAR Export window, select the **BapiCustomerApp** EAR project and the destination directory (the directory, including the EAR filename, where the project should be exported into).
4. Click **Finish**.

5. If the Save Resources window appears, click **OK**.

Now that you have exported the project to an Enterprise Application Archive (EAR) file, you are ready to install the application.

Installing the application

Installing the application project module is the last step of the deployment process. When you install the application and start it, the adapter, which is embedded in the project module, starts as part of the installed application.

Perform the following steps to install the application.

1. Before you install the application, you must create an authentication alias for use with your SAP instance. Once an authentication alias has been created, other SAP application project modules may use it as well.
   a. In the administrative console, click **Security** → **Global security**.
   b. On the right-side, under **Authentication**, click **JAAS Configuration** → **J2C Authentication data**.
   c. If an alias named SAP_Auth_Alias does not already exist, create it now.
   d. Click **New**. The General properties window appears.
   e. In the **Alias** field, specify `SAP_Auth_Alias`.
   f. Specify the userID and password for connecting to SAP.
   g. Click **OK**.
   h. Click **Save**.
2. To begin the installation process, click **Applications** → **Install New Applications**.
3. Under **Path to the new application**, specify the path of the EAR file, then click **Next**.
4. Continue to click **Next** through the various Step windows, until you reach the window titled with the step **Map resource reference to resources**.
5. Under **Specify authentication method**:
   a. Select the authentication alias you created earlier.
   b. Select the check box for the module.
   c. Click **Apply**.
6. Click **Next**. The Application Resource Warnings window appears.
7. Click **Continue**. The Install New Application window appears.
8. Click **Next**. The Ensure all unprotected 2.x methods have the correct level of protection window appears.
9. Click **Next**. A summary of all the installation options appears. Verify that all options are as you intended.
10. Click **Finish**.
11. A list of installation messages appears. Confirm that the message **Application installed successfully** is included at the end of the list.
12. Click the **Save to Master Configuration** link that appears at the end of the list of installation messages. The Enterprise Applications window appears.
13. Click **Save** to save the application. The application is now deployed and the Enterprise Applications window for the deployed application appears.
14. If the application is an inbound application, edit the J2C activation specification (ActivationSpec) properties. If the application is an outbound application, edit the J2C connection factory properties:

a. Click the deployed application and on the right-side column, under **Related Items**, click **Connector Module**. The Enterprise Application > Name of Application > Connector Modules window appears.
b. Click the name of the RAR file. The Enterprise Application > Name of Application > Connector Modules > Name of RAR file window appears.
c. Under **Additional properties**, click **Resource adapter**.
d. For inbound applications: Click **J2C Activation specifications** under **Additional properties**. For outbound applications, click **J2C connection factories** under **Additional properties**
e. For outbound applications: Click the factory instance created with the JNDI name specified in the EJB project.
f. Under **Additional properties**, click **Custom Properties**.
g. Update the desired property values.
h. For inbound applications only: Select the **Password** property you want to update, enter the password in the **Value** field, and click **OK**.
i. For inbound applications only: Click the **>** button to navigate to the next page and click the **userName** link.
j. For inbound applications only: Enter the username in the **Value** field and click **OK**.
k. Click the **Save** link in the **Messages** box at the top of the window.

15. Click the **Save** button to save the edits.

The application is now deployed and properly configured. The next step is to start the application.

Starting the application

After you have deployed the application, you can start it. Since the adapter is embedded in the application, when you start the application, the adapter is triggered to start running.

1. In the WebSphere Process Server administrative console, click **Applications** → **Enterprise Applications**.
2. Select the check box for the application and click the **Start** button. The application starts.

Configuring the adapter

To configure the adapter, you must set configuration properties.

Configuring properties

After you deploy the adapter, you can reconfigure adapter properties using the WebSphere Process Server administrative console.

You can configure the following properties using the administrative console:

- J2C connection factory properties (correspond to the ManagedConnectionFactory interface and are used for outbound processing)
- J2C activation specification properties (correspond to the ActivationSpec interface and are used for inbound processing)
- Custom properties (includes the default configuration properties of the adapter)

To configure properties using the administrative console, follow these steps.

1. Start the administrative console.

2. Under **Resources**, select **Resource Adapters**.
3. Under **Resource Adapters**, select **IBM SAP Adapter**.
   The General Properties page appears.
4. Under **Additional Properties**, select the category of properties you want to change:

Property Category	Description
J2C connection factories	To configure the ManagedConnectionFactory properties, which are used to configure a target Enterprise Information System (EIS) instance
J2C Activation specifications	To configure message endpoint properties.
Custom properties	To configure default configuration properties that are shared by all WebSphere Adapters.

5. Do one of the following:

Selection	Action
If you selected J2C connection factories	Select the name of the J2C connection factory you want to configure and then select **Connection pool properties**, **Advanced connection factory properties**, or **Custom properties**, depending on which J2C connection factory properties you want to configure. Custom properties are those J2C connection factory properties that are unique to the WebSphere Adapter for SAP Software. Connection pool and Advanced connection factory properties are properties you configure if you are developing your own adapter.
If you selected J2C Activation specifications	Select the name of the J2C activation specification that you want to configure. Then select the name of the message endpoint property you want to configure and set the value as desired.
If you selected Custom properties	The Custom properties page appears. Select the name of the default configuration property you wish to configure and set the value as desired.

Configuration properties of WebSphere Adapter for SAP Software

The WebSphere Adapter for SAP Software has several categories of configuration properties: J2C connection factory, J2C activation specification, resource adapter, and enterprise service discovery connection properties.

The following table describes the categories of adapter configuration properties.

Configuration property categories

Configuration property category	Description
J2C connection factory properties	Used to configure outbound processing and bidirectional enablement.
J2C activation specification properties	Used to configure inbound processing and bidirectional enablement.
Resource adapter properties	Used to configure features such as logging and tracing and bidirectional enablement.
Enterprise service discovery connection properties	Used during initial adapter deployment to configure either inbound or outbound processing and bidirectional enablement.

J2C connection factory properties

J2C connection factories configure an outbound connection to a target SAP instance. These properties correspond to the ManagedConnectionFactory interface of the J2EE Connector Architecture Specification.

A J2C connection factory manages connection pooling. It provides configuration information for outbound connectivity to a single SAP system instance from an application via the adapter.

The following table defines the SAP adapter-specific configuration properties that pertain to a J2C connection factory.

J2C connection factory properties

Property	Type	Globalized	Description
ABAPDebug	String	No	Specifies whether the adapter invokes the ABAP Debugger for the appropriate function module when the adapter begins processing a business object. When set to true, the adapter opens the ABAP Debugger. Debugging requires a dialog user with proper user authorizations. You can add breakpoints only after the debugger opens. **Important**: This property should always be set to false in a production environment. The default value is false.
ApplicationServerHost	String	Yes	When configuring the adapter to run without load balancing, specifies the IP address or the name of the application server that the adapter logs into. In both cases, the adapter assumes that the name of the gateway host is the same as the value specified for this property.
Client	Integer	No	SAP Client number under which the adapter logs in, often `100`.

Property	Type	Globalized	Description
Codepage	Integer	No	In order to establish a connection in the appropriate language, must correspond to the value specified in the Language property. For example, if Language is set to `JA` (Japanese), then Codepage must be set to `8000`, as dictated by the SAP application. Refer to SAP documentation for the exact Language and Codepage values.
EIS BiDi Format	String		The bi-di format used by SAP for its business data. The adapter normalizes the SAP bi-di data to the application server's logical left-to-right bi-di format for inbound communication and from the application server's format back to SAP bi-di format for outbound communication.
EIS Bidi Special Format	String		Signifies a category of values that are subject to special treatment during invocation of bi-di transformation to ensure accurate transformation of the category. Categories are predefined. For example: FTP URLs and email addresses.
GatewayHost	String	Yes	Host where the gateway service is running. (This property is set via the administrative console, and not during adapter deployment via the enterprise service discovery wizard.)
GatewayService	String	No	Gateway server identifier; often `sapgw00`. The system number of the server running the SAP Gateway (usually an application server) is 00. The value cannot be 00 if there is more than one server. The default is `sapgw00`. (This property is set via the administrative console, and not during adapter deployment via the enterprise service discovery wizard.)
Group	String	No	When configuring the adapter for load balancing, specifies the name of the logon group that represents a group of application servers. (This property is set via the administrative console, and not during adapter deployment via the enterprise service discovery wizard.)
Language	String	No	Language in which the adapter logs in. The default is `E`, for English.
MessageServerHost	String	Yes	When configuring the adapter for load balancing, specifies the name of the message server. (This property is set via the administrative console, and not during adapter deployment via the enterprise service discovery wizard.)

Property	Type	Globalized	Description
Password	String	Yes	Password for the adapter's user account on the SAP system. If bidirectional language support is enabled, this property is affected by the following BiDi properties, which are set using the enterprise service discovery wizard: • **Password BiDi Format**: Controls the bi-di format for this property. • **Skip BiDi Transformation for Password**: Controls invocation of bi-di transformation for this property.
Password BiDi Format	String		Controls the bi-di format for the Password property.
RFCTraceOn	boolean	No	Specifies whether or not to generate a text file detailing the RFC activity for each listener thread. You can specify a value of true or false. A value of true activates tracing, which generates a text file. Use these text files in a development environment only, because the files can grow rapidly. The default is false. (This property is set via the administrative console, and not during adapter deployment via the enterprise service discovery wizard.)
SAPSystemID	String	No	When configuring the adapter for load balancing, specifies the logical name of the SAP system, which is also known as `R3`*`name`*. (This property is set via the administrative console, and not during adapter deployment via the enterprise service discovery wizard.)
Skip BiDi Transformation	String		Controls invocation of bi-di transformation. Acceptable values: true or false. A blank value invokes the lookup mechanism.
Skip BiDi Transformation for Password	String		Controls invocation of bi-di transformation for the Password property. Acceptable values: true or false. A blank value invokes the lookup mechanism.
Skip BiDi Transformation for Username	String		Controls invocation of bi-di transformation for Username property. Acceptable values: true or false. A blank value invokes the lookup mechanism.
SystemNumber	Integer	No	System number of the application server. The value is a two-digit number, often 00. The default is 00.

Property	Type	Globalized	Description
Username	String	Yes	Name of the adapter's user account on the SAP system. If bidirectional language support is enabled, this property is affected by the following BiDi properties, which are set using the enterprise service discovery wizard: • **User Name BiDi Format**: Controls the bi-di format for this property. • **Skip BiDi Transformation for User Name**: Controls invocation of bi-di transformation for this property.
Username BiDi Format	String		Specifies the bi-di format for the Username property.

J2C activation specification properties

The J2C activation specification properties (also referred to as message endpoint properties) correspond to the ActivationSpec interface of the J2EE Connector Architecture Specification. An activation specification is a JavaBean used during endpoint activation. Endpoint activation is the process of notifying the adapter of eligible listener threads.

The following table defines the configuration properties that pertain to activating message endpoints.

J2C activation specification properties

Property	Type	Globalized	Description
AleFailureCode	Integer	No	Specifies the status code for dispatch failure. You must specify a value for this property (`68` or `58`) to cause the adapter to update the SAP failure status code after the adapter has retrieved an IDoc object for event processing. SAP converts this value to `40`.
AleFailureText	String	Yes	Specifies the descriptive text for dispatch failure. Specifying a value for this property is optional, even when you set AleUpdateStatus to true.

Property	Type	Globalized	Description
AleSelectiveUpdate	String	No	Specifies which IDoc Type and MessageType combinations are to be updated when the adapter is configured to update a standard SAP status code. You can define values for this property only if AleUpdateStatus has been set to true. The syntax for this property is: `IDocType: MessageType [;IDocType: MessageType [;...]]` where a semicolon (;) delimiter separates each IDoc Type and MessageType, and a comma (,) delimiter separates entries in a set. 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`.
AleStatusMsgCode	Integer	No	If required, specifies the message code to use when the adapter posts the ALEAUD Message IDoc (ALEAUD01). Configure this message code in the receiving Partner Profile. You can set a value for this property only if AleUpdateStatus has been set to true.
AleSuccessCode	Integer	No	Specifies the success status code for Application Document Posted. You must specify a value for this property (`52` or `53`) to cause the adapter to update the SAP success status code after the interface has retrieved an IDoc object for event processing. SAP converts this value to status 41 (Application Document Created in Receiving System).
AleSuccessText	String	Yes	Specifies the descriptive text for successful Application Document Posted. Specifying a value for this property is optional, even when you set AleUpdateStatus to true
AleUpdateStatus	boolean	No	Specifies whether an audit trail is required for all message types. This property must be set to true to cause the adapter to update a standard SAP status code after the adapter has retrieved an IDoc object for event processing.
ApplicationServerHost	String	Yes	When configuring the adapter to run without load balancing, specifies the IP address or the name of the application server that the adapter logs into.
AutoCreateEDT	boolean	No	Indicates whether the adapter should create the event recovery table automatically if it doesn't already exist. The default value is true.

Property	Type	Globalized	Description
BONameSpace	String		Namespace for the business object definitions to be used by this adapter. This value should be taken from the user-provided value during the metadata discovery process.
Client	Integer	No	SAP Client number under which the adapter logs in, often `100`.
Codepage	Integer	No	In order to establish a connection in the appropriate language, must correspond to the value specified in the Language property. For example, if Language is set to `JA` (Japanese), then Codepage must be set to `8000`, as dictated by the SAP application. Refer to SAP documentation for the exact Language and Codepage values.
EDT BiDi Format	String		Controls the BiDi format specific for all EDT properties.
EDTDatabaseName	String	No	Name of event recovery database. If bidirectional language support is enabled, this property is affected by the following bi-di properties, which are set using the enterprise service discovery wizard: • **EDT BiDi Format**: Controls the bi-di format for all EDT properties. • **Skip BiDi Tranformation for EDT**: Controls invocation of bi-di transformation for EDT properties.
EDTDriverName	String	No	Name of the XA database driver used to connect to the Event Recovery Table for inbound events. Example: `com.ibm.db2j.jdbc. DB2jXADataSource` (for Cloudscape).
EDTSchemaName	String	No	Schema used for auto-creating the event recovery database. If bidirectional language support is enabled, this property is affected by the following bi-di properties, which are set using the enterprise service discovery wizard: • **EDT BiDi Format**: Controls the bi-di format for all EDT properties. • **Skip BiDi Tranformation for EDT**: Controls invocation of bi-di transformation for EDT properties.

Property	Type	Globalized	Description
EDTTableName	String	No	Name of event recovery table. If bidirectional language support is enabled, this property is affected by the following bi-di properties, which are set using the enterprise service discovery wizard: • **EDT BiDi Format**: Controls the bi-di format for all EDT properties. • **Skip BiDi Transformation for EDT**: Controls invocation of bi-di transformation for EDT properties.
EDTURL	String	No	The URL to the EDT database. If bidirectional language support is enabled, this property is affected by the following bi-di properties, which are set using the enterprise service discovery wizard: • **EDT BiDi Format**: Controls the BiDi format for EDTURL. • **EDT URL BiDi Special Format**: Specifies the category of cases subject to special treatment during invocation of bi-di transformation. • **Skip BiDi Transformation for EDT URL**: Controls invocation of bi-di transformation for EDTURL.
EDT URL BiDi Special Format	String		Specifies the category of cases subject to special treatment during invocation of bi-di transformation to ensure accurate transformation of the category.
EDTUserName	String	Yes	User name for connecting to the database. If bidirectional language support is enabled, this property is affected by the following bi-di properties, which are set using the enterprise service discovery wizard: • **EDT BiDi Format**: Controls the BiDi format for all EDT properties. • **Skip BiDi Transformation for EDT**: Controls invocation of BiDi transformation for EDT properties.
EDTUserPassword	String	Yes	User password for connecting to the database. If bidirectional language support is enabled, this property is affected by the following bi-di properties, which are set using the enterprise service discovery wizard: • **EDT BiDi Format**: Controls the BiDi format for all EDT properties. • **Skip BiDi Transformation for EDT**: Controls invocation of BiDi transformation for EDT properties.

Property	Type	Globalized	Description
EIS BiDi Format	String		The bi-di format used by SAP for its business data. The adapter normalizes the SAP bi-di data to the application server's logical left-to-right bi-di format for inbound communication and from the application server's format back to SAP bi-di format for outbound communication.
EIS BiDi Special Format	String		Specifies the category of values that are subject to special treatment during invocation of bi-di transformation to ensure accurate transformation of the category. Categories are predefined. For example: FTP URLs and email addresses.
GatewayHost	String	Yes	SAP Gateway Host where the gateway service is running.
GatewayService	String	No	Gateway server identifier; often `sapgw00`. `00` is the system number of the server running the SAP Gateway (usually an application server). May not be `00` if you have more than one server. The default is `sapgw00`.
Group	String	No	When configuring the adapter for load balancing, specifies the name of the logon group that represents a group of application servers.
IgnoreIDocPacketErrors	boolean	No	If the adapter encounters an error while processing the IDoc packet, it can behave in two different ways depending upon the `IgnoreIDocPacketErrors` configuration property. When this property is set to false, the adapter stops processing further IDocs in that packet and reports an error to the SAP system. When this property is set to true, the adapter logs an error and continues processing the rest of the IDocs in that packet.
Language	String	No	Language in which the adapter logs in. The default is `E`, for English.
MessageServerHost	String	Yes	When configuring the adapter for load balancing, specifies the name of the message server.

Property	Type	Globalized	Description
NumberOfListeners	Integer		Specifies the number of listener threads that are created when the adapter is initialized. A listener thread can handle one request at a time. Each listener thread handles a single event at a time. If you have multiple listener threads, the adapter can handle multiple events concurrently. The default is 1. It is recommended that you have no more listener threads than the available work processes in SAP.
Password	String	Yes	Password for the adapter's user account on the SAP system. If bidirectional language support is enabled, this property is affected by the following bi-di properties, which are set using the enterprise service discovery wizard: • **Password BiDi Format**: Controls the BiDi format for this property. • **Skip BiDi Transformation for Password**: Controls invocation of BiDi transformation for this property.
RfcProgramID	String	Yes	Program identifier that the RFC Server program registers under.
RFCTraceOn	boolean	No	Specifies whether or not to generate a text file detailing the RFC activity for each listener thread. You can specify a value of true or false. A value of true activates tracing, which generates a text file. Use these text files in a development environment only, because the files can grow rapidly. The default value is false.
SAPSystemID	Integer	No	When configuring the adapter for load balancing, specifies the logical name of the SAP system, also known as R3*name*.
Skip BiDi Transformation	String		Controls invocation of bi-di transformation. Acceptable values: true or false. A blank value invokes the lookup mechanism.
Skip BiDi Transformation for EDT	String		Controls invocation of bi-di transformation for EDT properties. Acceptable values: true or false. A blank value invokes the lookup mechanism.
Skip BiDi Transformation for EDT URL	String		Controls invocation of bi-di transformation for EDT URL. Acceptable values: true or false. A blank value invokes the lookup mechanism.

Property	Type	Globalized	Description
SplitIDocPackets	boolean	No	Indicates whether or not the adapter needs to send the entire IDoc packet or split it into multiple IDocs.
SystemNumber	String	No	System number of the application server. The value is a two-digit number, often `00`. The default value is `00`.
Username	String	Yes	Name of the adapter's user account on the SAP system. If bidirectional language support is enabled, this property is affected by the following bi-di properties, which are set using the enterprise service discovery wizard: • **User Name BiDi Format**: Controls the bi-di format for this property. • **Skip BiDi Transformation for User Name**: Controls invocation of BiDi transformation for this property.

Resource adapter properties

Using the enterprise service discovery wizard when you first configure the adapter (and later, via the WebSphere Process Server administrative console), you can configure Resource Adapter properties. This category of properties includes logging and tracing properties, bidirectional properties, and adapter-specific properties.

Logging and tracing properties

The following table describes the logging and tracing properties for the adapter.

Logging and tracing properties

Property	Type	Description
LogFileName	String	The full path of the log file. This property is required.
LogNumberOfFiles	Integer	The number of log files to use. When a log file reaches its maximum size the adapter will start using another log file. If no value is specified, it will be set to 1. This property is not required.
LogMaxFileSize	Integer	Size of the log files in kilobytes. If no value is specified, the file will have no maximum size. This property is not required.
TraceFileName	String	The full path to the trace file. This property is required.
TraceNumberOfFiles	Integer	The number of trace files to use. When a trace file reaches its maximum size the adapter will start using another trace file. If no values is specified, it will be set to 1. This property is not required.
TraceFileSizeMax	Integer	Size of the trace files in kilobytes. If no value is specified, the file will have no maximum size. This property is not required.

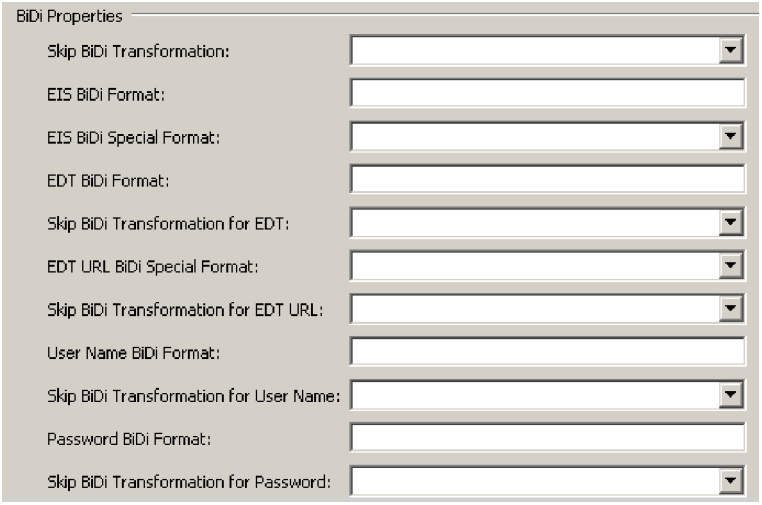
Bidirectional properties

The following table describes the bidirectional properties for the adapter. These properties display in the last window of the enterprise service discovery wizard and are used to define bi-di formats for the adapter at runtime. These properties differ from the bidirectional properties that appear in the *first* window of the enterprise service discovery wizard, which are used to define bi-di formats for the enterprise service discovery wizard.

Bidirectional properties

Property	Type	Description
EIS BiDi Format	String	The bi-di format used by SAP for its business data. The adapter normalizes the SAP bi-di data to the application server's logical left-to-right bi-di format for inbound communication and from the application server's format back to SAP bi-di format for outbound communication.
Skip BiDi Transformation	String	Controls invocation of bi-di transformation. Acceptable values: true or false. A blank value invokes the lookup mechanism.
EIS BiDi Special Format	String	Signifies a category of values that are subject to special treatment during invocation of bi-di transformation to ensure accurate transformation of the category. Categories are predefined. For example: FTP URLs and email addresses.
Turn BiDiOff	Boolean	A flag used to turn off (explicitly exclude) bi-di support. This property takes precedence over the BiDiSkip property, and it allows users who do not require bidirectional script data support to turn it off.

The following screen illustrates the portion of the enterprise service discovery wizard window in which these properties appear:



Bidirectional properties on the last window of the enterprise service discovery wizard

Adapter-specific property

The following table describes the adapter-specific configuration property that is unique to the WebSphere adapter for SAP Software.

Adapter-specific property

Property	Type	Globalized	Description
PartnerCharset	String	No	Specifies a PartnerCharset encoding. When populated with an encoding value, the value provided is used for the data conversion. If no value is provided, the value is obtained from the client connection to the SAP system

Enterprise service discovery connection properties

The enterprise service discovery wizard requires that you set properties when you initially deploy the adapter. The first window of the enterprise service discovery window (the Configure Settings for Discovery Agent window) is used to configure the following categories of properties: logon connection properties required for connecting to the SAP application to perform metadata discovery, metadata properties, and bidirectional configuration properties.

Logon connection properties

The following table describes the logon connection properties required by the enterprise service discovery wizard to log on to the SAP application.

Logon connection properties

Property	Type	Globalized	Description
ApplicationServerHost	String	Yes	When configuring the adapter to run without load balancing, specifies the IP address or the name of the application server that the adapter logs into. In both cases, the adapter assumes that the name of the gateway host is the same as the value specified for this property.
Client	Integer	No	SAP Client number under which the adapter logs in, often `100`.
CodepageNumber	Integer	No	In order to establish a connection in the appropriate language, must correspond to the value specified in the Language property. For example, if Language is set to `JA` (Japanese), then CodepageNumber must be set to `8000`, as dictated by the SAP application. Refer to SAP documentation for the exact Language and Codepage values.
Language	String	No	Language in which the adapter logs in. The default is `E`, for English.
Password	String	Yes	Pass of the adapter's user account on the SAP system.

Property	Type	Globalized	Description
RFCTraceOn	boolean	No	Specifies whether or not to generate a text file detailing the RFC activity for each listener thread. You can specify a value of true (checked) or false (unchecked). A value of true activates tracing, which generates a text file. Use these text files in a development environment only, because the fields can grow rapidly. The default value is false (unchecked).
SystemNumber	Integer	No	System number of the application server. The value is a two-digit number, often 00. The default is 00.
UserName	String	Yes	Name of the adapter's user account on the SAP system.

Metadata properties

The following table describes the metadata properties required by the enterprise service discovery wizard to introspect and display the correct elements from the SAP application.

Metadata properties

Property	Type	Description
Select the Module	String	Indicates whether you are creating business objects for the ALE or BAPI interface.
Maximum number of hits for the discovery	Integer	When introspecting elements in the ALE or BAPI interface, defines the maximum number of SAP elements displayed by the wizard per discovery. Default value is 100.

Bidirectional properties

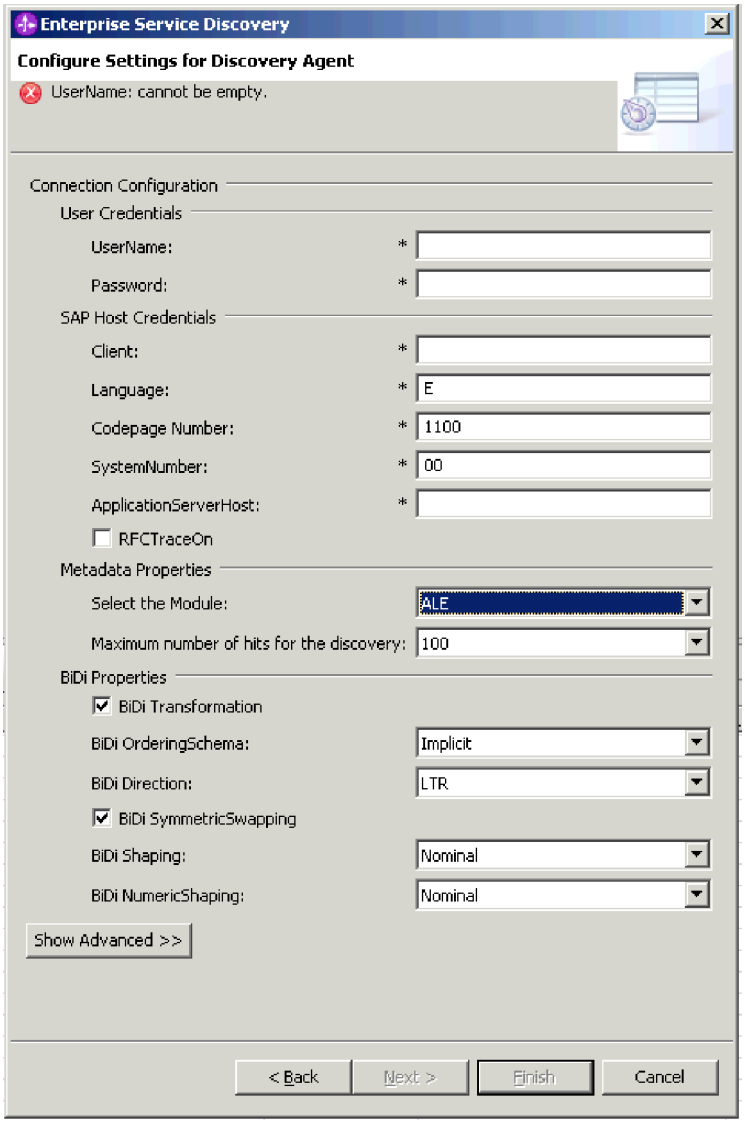
When you initially configure the adapter, if you want to enable bidirectional language support for the enterprise service discovery wizard, you must configure a set of six properties. These properties determine the bi-di format of field names that display throughout the enterprise service discovery wizard and the bi-di format of the values you enter in these fields. The properties, which are used by the enterprise service discovery wizard during adapter configuration to communicate with the SAP application, appear in the first window of the wizard (the Configure Settings for Discovery Agent window) under the heading **BiDi properties**. Note that the only fields in this first window that are affected by the bi-di format defined are User Name and Password. Other fields in this first window do not have bi-di support.

The following table describes the **BiDi properties** that appear in the first window of the enterprise service discovery wizard. Note that these differ from bidirectional properties that appear in the *last* window of the enterprise service discovery wizard, in that the following properties define bi-di for the enterprise service discovery wizard, whereas the ones on the last enterprise service discovery wizard window define bi-di formats for the adapter.

Bidirectional properties (first window of the enterprise service discovery wizard)

Property	Type	Description
BiDiTransformation	Boolean	Turns bidirectional support on or off. The default value is false, which means that bi-di support is turned off.
BiDOrderingSchema	String	Determines the type of text schema used; either implicit (logical), or visual. The default value is implicit.
BiDiDirection	String	Determines the text direction used. Possible values are LTR (left to right), RTL (right to left), ContextualLTR (contextual left to right), and ContextualRTL (contextual right to left.) The default value is LTR.
BiDiSymmetricSwapping	Boolean	Determines whether systemic swapping is turned on or off. The default value is True, which means that systemic swapping is turned on.
BiDi Shaping	String	Determines the bi-di format used by the enterprise service discovery wizard while it communicates with the SAP application. Possible values are Nominal, National, Contextual The default value is Nominal.
BiDiNumericShaping	Sting	Determines the bi-di format used by the enterprise service discovery wizard while it communicates with the SAP application. Possible values are Nominal, National, Contextual The default value is Nominal.

The following image illustrates the enterprise service discovery wizard window in which these properties appear.



First window of the enterprise service discovery wizard

Troubleshooting the adapter

This topic introduces the task of troubleshooting the WebSphere Adapter for SAP Software.

Contacting IBM Software Support

IBM Software Support provides assistance with product defects.

Before contacting IBM Software Support, your company must have an active IBM software maintenance contract, and you must be authorized to submit problems to IBM. The type of software maintenance contract that you need depends on the type of product you have:

- For IBM distributed software products (including, but not limited to, Tivoli, Lotus®, and Rational® products, as well as DB2 and WebSphere products that run on Windows or UNIX operating systems), enroll in Passport Advantage® in one of the following ways:
  - **Online:** Go to the Passport Advantage Web page and click How to Enroll.

  - **By phone:** For the phone number to call in your country, go to the contacts page of the IBM Software Support Handbook on the Web and click the name of your geographic region.
- For IBM eServer™ software products (including, but not limited to, DB2 and WebSphere products that run in zSeries®, pSeries®, and iSeries™ environments), you can purchase a software maintenance agreement by working directly with an IBM sales representative or an IBM Business Partner. For more information about support for eServer software products, go to the IBM Technical Support Advantage Web page.

If you are not sure what type of software maintenance contract you need, call 1-800-IBMSERV (1-800-426-7378) in the United States or, from other countries, go to the contacts page of the IBM Software Support Handbook on the Web and click the name of your geographic region for phone numbers of people who provide support for your location.

To contact IBM Software support, follow these steps:

- Determine the business impact of your problem.
- Describe your problem and gather background information.
- Submit your problem to IBM Software Support.

1. Determine the business impact of your problem. When you report a problem to IBM, you will be asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem you are reporting. Use the following criteria:

Severity	Description
Severity 1	Critical business impact: You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.
Severity 2	Significant business impact: The program is usable but is severely limited.
Severity 3	Some business impact: The program is usable with less significant features (not critical to operations) unavailable.
Severity 4	Minimal business impact: The problem causes little impact on operations, or a reasonable circumvention to the problem has been implemented.

2. Describe your problem and gather background information. When explaining a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you solve the problem efficiently. To save time, know the answers to these questions:
   - What software versions were you running when the problem occurred?
   - Do you have logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
   - Can the problem be recreated? If so, what steps led to the failure?
   - Have any changes been made to the system? (For example, hardware, operating system, networking software, and so on.)
   - Are you currently using a workaround for this problem? If so, please be prepared to explain it when you report the problem.
3. Submit your problem to IBM Software Support. You can submit your problem in one of two ways:
   - **Online:** Go to the Submit and track problems page on the IBM Software Support site. Enter your information into the appropriate problem submission tool.

- **By phone:** For the phone number to call in your country, go to the contacts page of the IBM Software Support Handbook on the Web and click the name of your geographic region.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support will create an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail.

Whenever possible, IBM Software Support will provide a workaround for you to implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM product support Web pages daily, so that other users who experience the same problem can benefit from the same resolutions.

Enabling logging

The WebSphere Adapter for SAP Software maintains a log file that you can view to determine the status of event processing. All events and errors that relate to the adapter are tracked by the log file, along with the date, time, and event for each log entry. Since the adapter logs an error message when it encounters an error or warning condition, the log file is a good source to start troubleshooting problems.

Logging for the adapter is enabled through the WebSphere Process Server administrative console. Follow the steps below to enable logging.

1. Start the WebSphere Process Server administrative console.
2. From the Administrative Console, select **Troubleshooting → Logs and Trace**.
3. Click **Component** to specify a log detail level for individual components or click **Groups** to specify a log detail for a predefined group of components.
4. Select the logging level that you need. The "Logging levels" table describes the different logging levels that can be set for the adapter through the WebSphere Process Server administrative console. These adapter log levels are the same as those that you set for the enterprise service discovery wizard.

   Note: To view log events that are below the Detail Level, you must enable the Diagnostic Trace Service. Log events that are at Detail Level or above can be viewed in the SystemOut log, the IBM Service log (when enabled), or the Diagnostic Trace Service (when enabled).

   Logging levels

Level	Indicator	Description
Audit	A	Significant event affecting server state or resources
Config	C	Configuration change or status.
Detail	D	General information detailing subtask progress.
Fatal	F	Task cannot continue. Component cannot function.
Info	I	General information outlining overall task progress
Severe	E	Task cannot continue. Component can still function. This also includes conditions that indicate an impending fatal error - i.e. reporting on situations that strongly suggest that resources are on the verge of being depleted.
Warning	W	Potential error or impending error. This also includes conditions that indicate a progressive failure - for example, the potential leaking of resources.

5. Click **Apply** to save your changes.

Enabling tracing

Tracing determines what level of errors or warnings are captured in the adapter log file. You can trace messages regarding adapter processing by defining a tracing level. Trace messages, by default, are captured in the adapter log file, but can be captured in a separate file if you prefer. (For the enterprise service discovery wizard, the trace messages are captured in the same file as that used for the wizard's log messages.)

The trace levels can be configured in the WebSphere Process Server administrative console. Follow the steps below to enable and set tracing levels.

1. Start the WebSphere Process Server administrative console.
2. From the administrative console, select **Troubleshooting → Logs and Trace**.
3. Select the Tracing level that you need. The "Tracing levels" table describes the different tracing levels that can be set through the WebSphere Process Server administrative console.

   Tracing levels

Level	Indicator	Description
Fine	1	General trace. Includes broad actions being taken by adapter such as establishing a connection to the EIS, converting an event in the EIS to a business object (only key values), processing a business object (only key values).
Finer	2	Detailed trace that provides more granular information on the logic being performed by the adapter including the various API calls being made to the EIS and any parameters or return values.
Finest	3	This is the most detailed level and should include method entry / exit / return values. Complete business object dumps should be included. At this level, all detail needed to debug problems should be provided.

4. Click **Apply** to save your changes.

Enabling the Common Event Infrastructure (CEI)

This topic describes how to enable the Common Event Infrastructure (CEI) for the adapter.

You must publish the IBM WebSphere Adapters Event Definitions file to the CEI catalog before you can set these event definitions. For instruction on how to do this, refer to the CEI documentation found on the WebSphere Process Server web site at http://www.ibm.com/software/integration/wps.

1. Start the WebSphere administrative console.
2. Go to **Troubleshooting → Log and Trace** and select <your server name>.
3. There are many options for the General Properties. Select **Change Log Detail Level**, and then select **com.ibm.j2ca.*** for JCA components. Under this section there is a subcomponent for each adapter type:
   - com.ibm.j2ca.flatfile.* (WebSphere Adapter for Flat Files)
   - com.ibm.j2ca.jdbc.* (WebSphere Adapter for JDBC)
   - com.ibm.j2ca.peoplesoft.* (WebSphere Adapter for PeopleSoft)
   - com.ibm.j2ca.sap.* (WebSphere Adapter for SAP)
   - com.ibm.j2ca.siebel.* (WebSphere Adapter for Siebel)

4. Select the component that matches your adapter. Each adapter component has two subcomponents, one for logging and one for CEI. They are:
   - *subcomponent name*.log.*adapter id*
   - *subcomponent name*.cei.*adapter id*

   For example, com.ibm.j2ca.siebel.cei.<AdapterID1>. For each instance of a deployed adapter, the system will show a separate ID.
5. Select the CEI adapter ID that you want to enable.
6. From the drop-down menu, you can choose from the following:
   - off - turn CEI off
   - fine- turn CEI on with Event Content set to Empty
   - finer- turn CEI on with Event Content set to Digest
   - finest- turn CEI on with Event Content set to Full
   - all - same as finest

For information on what each Event Content level means (Empty, Digest and Full), and for more information on using the Common Base Event model and the Common Event Infrastructure, refer to the documentation on the WebSphere Process Server web site at http://www.ibm.com/software/integration/wps

Using the sample applications

The adapter provides three sample applications that illustrate how you deploy an application package and how the adapter processes business objects. Each application presents two scenarios, one for each audience of the adapter. The adapter audience consists of two users: the application integrator and the data integrator.

For each of the samples provided, the two scenarios presented are as follows:

Sample application scenarios

Scenario	Description	Audience
Scenario 1	• Provides the already-generated artifacts and illustrates how the adapter processes business objects. Using the Enterprise Service Discovery to generate artifacts is not required in this scenario. • Targeted at an audience that is responsible for assembling application components into a solution and preparing this solution for testing and deployment.	Application integrator
Scenario 2	• Illustrates how you use the Enterprise Service Discovery tool to discover SAP application components and develop the business objects that the adapter processes. • Targeted at an audience with the same responsibilities as the application integrator, but is further responsible for enabling access to a range of data sources for the application developers.	Data integrator

The samples that are included illustrate the following scenarios:

- BAPI outbound processing
- ALE outbound processing
- ALE inbound processing

Adapter dependencies

Before running the samples, make sure you have added the external dependencies correctly. This includes copying the SAP Java API (sapjco.jar file) and dependencies libraries to the correct WebSphere Process Server directories, as described in the deployment information.

BAPI outbound sample application

The BAPI outbound sample application illustrates how the WebSphere Adapter for SAP Software creates business objects based on various BAPI customer-related function calls.

The sample shows how to configure the adapter as an SCA component, and, once deployed how to invoke it to create a Customer object in SAP by invoking the BAPI.

Structure of the application package

The sample application files are installed when you install the adapter. They are packaged in an archive file that is installed in the samples folder.

Scenario 1: All-inclusive set of sample files

For Scenario 1, targeted at the Application Integrator, the sample application package includes all the required artifacts, so you do not need to use the enterprise service discovery wizard to obtain these. In a real-world deployment of the adapter, you would generate all these artifacts and configure the adapter using the enterprise service discovery wizard, as illustrated in Scenario 2

The files for Scenario 1 are archived in the following file: \samples\ nonemdsample.zip.

Uncompress this .zip file to extract the file \BAPI\BapiCustomerApp.ear to the location of your choice. This filename is based on the module name specified in the enterprise service discovery wizard.

The following files are extracted from the .EAR file:

- A configured instance of the adapter that is deployed by default to your local host. The file name is CWYAP_SAPAdapter.rar.
- An SCA module with the following SCA artifacts:
  - BapiCustomerEJB.jar
  - BapiCustomer.jar
  - BapiCustomerEJBClient.jar
  - BapiCustomerWeb.war

Scenario 2: Enterprise Service Discovery requirement

For Scenario 2, targeted at the Data Integrator, the sample application package provides the capability for you to create the SCA artifacts and configure the adapter using the enterprise service discovery wizard. You can, however, access copies all the files that you eventually generate using the Enterprise Service

Discovery, as a way of verifying that what you create with the Enterprise Service Discovery is indeed correct and what the application expects in order to run properly.

All the necessary files, including copies of those you will generate with the enterprise service discovery wizard (for verification after you create your own copies) are archived in the following file: \samples\emdsample.zip.

Uncompress this file to extract the following files for Scenario 2:

- SAPOutboundInterface.import
- SAPOutboundInterface.wsdl
- sca.module
- SapBapiCustomerChangefromdata.xsd
- SapBapiCustomerCreatefromdata.xsd
- SapBapiCustomerCreateWrapper.xsd
- SapBapiCustomerCreateWrapperBG.xsd
- SapBapiCustomerGetDetail.xsd
- SapPeAddress.xsd
- SapPeAddress963341981.xsd
- SapPiAddress.xsd
- SapPiAddress655351271.xsd
- SapPiCopyreference.xsd
- SapReturn.xsd
- SapReturn619647890.xsd

Business object structure

The structure of the business objects used in the BAPI outbound sample is based on the metadata of the BAPI_CUSTOMER_CREATEFROMDATA, BAPI_CUSTOMER_CHANGEFROMDATA, and BAPI_CUSTOMER_GETDETAIL calls.

Deploying and configuring for Scenario 1 (BAPI outbound)

Scenario 1 of the BAPI outbound sample application provides a configured instance of the adapter and all the necessary SCA artifacts, so you are not required to deploy the package and configure the adapter. Simply extract the .EAR file and import the RAR file to your WebSphere Integration Developer workspace using WebSphere Integration Developer.

After you uncompress the nonemdsample.zip file, perform the following steps to extract the contents of BapiCustomerApp.ear and to import the RAR file.

1. Uncompress the file BapiCustomerApp.ear to a directory of your choice. The CWYAP_SAPAdapter.rar file and various JAR files are now listed in the directory.
2. If the connector project is not already defined in your WebSphere Integration Developer workspace, import the latest CWYAP_SAPAdapter.rar file in WebSphere Integration Developer. Importing the RAR file creates a connector project for the adapter.
3. In WebSphere Integration Developer, create a module and assign it the same name as the EAR file without the letters “App.” For example, if the EAR file is BapiCustomerApp.ear, then the module name in this case should be `BapiCustomer`.

4. Extract the contents of the module JAR file into the module. If prompted, do **not** overwrite the .classpath or .runtime files. For example, extract BapiCustomer.jar into the module you just created called `BapiCustomer`.
5. In WebSphere Integration Developer, switch to the Business Integration perspective, right-click on the module, and select **Refresh** from the pop-up menu. You should now see all the .xsd files in the **Data Types** folder.
6. Do the following to associate the module with the SAP connector project:
   a. Right-click the module and select **Properties** from the pop-up menu.
   b. Select **Java build path** from the left-side menu.
   c. Click the Projects tab and select the check box for the connector project you created in 2 on page 87
7. Do the following to associate the BapiCustomerApp project with the SAP connector project.
   a. In the Java perspective, right-click the BapiCustomerApp project.
   b. Select **Project References** from the left-side menu.
   c. Select the check box for the connector project you created in 2 on page 87.
   d. Within the BapiCustomerApp project, locate the file application.xml and open it using Deployment Descriptor Editor.
   e. In the **Modules** section, click **Details**.
   f. On the next panel, click on **Add** and add CWYAP_SAPAdapter.rar.
   g. Press **Ctrl**+**S** to save your changes.
8. Select **Project** → **Clean** from the WebSphere Integration Developer menu to rebuild the projects.
9. Start up WebSphere Process Server.
10. Add the project to WebSphere Process Server.
11. Go to the administrative console and ensure that the application has successfully started running.
12. If any processes are not running properly, stop and restart the server.
13. Ensure that the SAP_Auth_Alias has been properly configured with the userID and password required to log on to the SAP application.

The adapter configuration properties are already set to the appropriate values in the adapter instance that is provided with the sample. However, you must change the values of the following properties to suit your configuration:

Properties to set for your configuration

Property	Type	Globalized	Description
ApplicationServerHost	String	Yes	When configuring the adapter to run without load balancing, specifies the IP address or the name of the application server that the adapter logs into. In both cases, the adapter assumes that the name of the gateway host is the same as the value specified for this property.
Password	String	Yes	Password for the adapter's user account on the SAP system.
Username	String	Yes	Name of the adapter's user account on the SAP system.

To change these values, use the WebSphere Process Server Administrative Console.

Deploying and configuring for Scenario 2 (BAPI outbound)

Scenario 2 of the BAPI outbound sample application requires you to use the enterprise service discovery wizard to deploy the application package, configure the adapter, and generate the SCA artifacts.

Before you begin to deploy and configure:

- Import the CWYAP_SAPAdapter.RAR file into the project.
- Add the external dependencies (SAP Java Connector (SAP JCo) interface) to the project.

For this task, you must run the Enterprise Service Discovery within IBM WebSphere Integration Developer to set adapter configuration properties. To change the property values later, use the WebSphere Process Server Administrative Console.

1. In the Business Integration Perspective of WebSphere Integration Developer, right-click the frame and from the pop-up menu, select **New → Enterprise Service Discovery**.
2. In the Import Configurations window, select **IBM WebSphere Adapter for SAP Software**.
3. In the Discovery Agent Initialize Properties window, specify the configuration properties to initialize the discovery agent. Be sure to set the **Select Module** field to the value of `BAPI`. Properties marked with an asterisk (*) are mandatory.
4. At the bottom of the Discovery Agent Initialize Properties window, click the **Show Advanced** button.
5. When prompted to specify logging options, set the **Logging Level** to `FINEST` to provide the highest level of logging and then click **Next**.
6. In the Metadata Query window, click **Run Query**.
7. Under **Objects Discovered**, drill down to the **Discover By Name** node, and then click the **Filter** button.
8. In the Filter Properties for Discover by Name window, enter `BAPI_CUSTOMER*`, 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*.
9. Click **OK**.
10. Navigate to BAPI_CUSTOMER_CREATFROMDATA and click the **Add** button.
11. In the Configuration Parameters window, select the **Use Field Name to generate attribute(s)** check box, and click **OK** to add BAPI_CUSTOMER_CREATFROMDATA to the list of business objects to be imported.
12. Repeat steps 10 and 11 for BAPI_CUSTOMER_CHANGEFROMDATA. The selected objects appear in the bottom frame, under **Objects to be imported**. Objects can be removed from the list by selecting the object and clicking the **Remove** button.
13. Click **Next**.
14. In the Selection Properties window, enter `BapiCustomerCreate` as the name of the business object and specify the appropriate JCo methods for each of the object's operations. For example, specify the following methods:
    - For the CREATE operation, specify `BAPI_CUSTOMER_CREATEFROMDATA`.

- For the UPDATEWITHDELETE operation, specify `BAPI_CUSTOMER_CHANGEFROMDATA`.
- For the RETRIEVE operation, specify `BAPI_CUSTOMER_GETDETAIL`.

15. Click **Next**.
16. In the Generate Artifacts window, specify the module name were the artifacts should be saved and then click **New** to create a new business integration module.
17. When prompted, enter `BapiCustomer` as the module name and click **OK**.
18. Specify `ScaArtifacts` as the folder name within the module where the Service Description should be saved.
19. In **J2C Authentication Data Entry**, enter `SAP_Auth_Alias` and select the **Deploy connector with module** check box.
20. Select the **Use discovered connection properties** radio button, enter values for the connections properties, and click **Finish**. The new BapiCustomer module is added to the Business Integration Perspective, along with all its artifacts.
21. Export the project to the .EAR file (note that the .EAR file does not yet exist; this step defines it):
    a. Right-click the BapiCustomer module and select **Export**.
    b. Select **EAR file** from the Export window.
    c. Click **Next**.
    d. In the EAR Export window, select `BapiCustomerApp` for the .EAR project and specify its destination directory. Then click **Finish**.
22. In the EAR Export window, export the EAR file of the adapter:
    a. Select the EAR file in the **EAR project** field.
    b. Specify its destination directory in the **Destination** field.
    c. Click **Finish**.
23. Deploy the .EAR file.

The adapter application package, including all its SCA artifacts, is now configured and deployed. You can now run the sample application.

Running the sample application

After you deploy and configure the sample application package and adapter, you can run the application to illustrate how the adapter supports outbound processing of BAPI business objects.

1. Start the administrative console and ensure that the BAPI outbound sample application has been installed.
2. Ensure that the connection factory properties are correctly configured and that the J2C authentication alias, SAP_Auth_Alias, has the proper credentials to log onto the SAP system.
3. Switch to the Business Integration perspective in WebSphere Integration Developer.
4. To start testing, right-click the name of the module and select **Test → Module**. The WebSphere Business Integration Test Client displays a panel allowing you to select the Configuration, Module, Component, Interface and Operation you wish to test.
5. For the BAPI outbound sample application, select the following values:
   - **Configuration**: `Default Module Test`
   - **Module**: `BapiCustomer`
   - **Component**: `SAPOutboundInterface`

- **Interface**: `SAPOutboundInterface`
- **Operation**: `createSapBapiCustomerWrapper`

6. In the Initial Request Parameters window, enter the verb and the value(s) you want to send to the adapter. Possible values for the verb are: `Create`, `Delete` and `UpdateWithDelete`.
7. Press **Continue**. The Select Deployment Location window appears.
8. Select the WebSphere Process Server instance on which you want to test and click **Finish**.
   - The Starting the integration test client window appears.
   - The Return parameters window displays with the data object returned by the adapter.

ALE outbound sample application

The ALE outbound sample application illustrates how the WebSphere Adapter for SAP Software creates a customer master IDoc called DEBMAS03 in the SAP system.

The sample shows how to configure the adapter as an SCA component, and, once deployed how to create the IDoc. It does not show how to process the IDoc in the SAP system to create a customer master record.

Structure of the application package

The sample application files are installed when you install the adapter. They are packaged in an archive file that is installed in the samples folder.

Scenario 1: All-inclusive set of sample files

For Scenario 1 of the ALE outbound sample application, targeted at the Application Integrator, the sample application package includes all the required artifacts, so you do not need to use the enterprise service discovery wizard to obtain these. In a real-world deployment of the adapter, you would generate all these artifacts and configure the adapter using the enterprise service discovery wizard, as illustrated in Scenario 2.

The files for Scenario 1 are archived in the following file: \samples\ nonemdsample.zip.

Uncompress this file to extract the file \ALE\CustomerMasterIDocApp.ear to the location of your choice.

The following files are extracted from the .EAR file:

- A configured instance of the adapter that is deployed by default to your local host. The file name is CWYAP_SAPAdapter.rar.
- An SCA module with the following SCA artifacts:
  - CustomerMasterIDoc.jar
  - CustomerMasterIDocEJB.jar
  - CustomerMasterIDocEJBClient.jar
  - CustomerMasterIDocWeb.war

Scenario 2: Enterprise service discovery wizard requirement

For Scenario 2 of the ALE outbound sample application, targeted at the Data Integrator, the sample application package provides the capability for you to create

the SCA artifacts and configure the adapter using the enterprise service discovery wizard. You can, however, access copies of all the files that you eventually generate using the enterprise service discovery wizard, as a way of verifying that what you create with the tool is indeed correct and what the application expects in order to run properly.

All the necessary files, including copies of those you will generate with the enterprise service discovery wizard (for verification after you create your own copies) are archived in the following file: \samples\emdsample.zip.

Uncompress this .zip file to extract the following files for Scenario 2 to the samples directory:

- SAPOutboundInterface.import
- SAPOutboundInterface.wsdl
- sca.module
- SapDebmas03.xsd
- SapDebmas03BG.xsd
- SapDebmas03DataRecord.xsd
- SapDebmas03E2kna11001356232861.xsd
- SapDebmas03E2kna1h001354594356.xsd
- SapDebmas03E2kna1l824375735.xsd
- SapDebmas03E2kna1m004354445398.xsd
- SapDebmas03E2knasm824373688.xsd
- SapDebmas03E2knb1h001325965205.xsd
- SapDebmas03E2knb1l824374774.xsd
- SapDebmas03E2knb1m005325816246.xsd
- SapDebmas03E2knb5m824374649.xsd
- SapDebmas03E2knbkm002272252031.xsd
- SapDebmas03E2knexm824369689.xsd
- SapDebmas03E2knkam824364636.xsd
- SapDebmas03E2knkkh00114738628.xsd
- SapDebmas03E2knkkl824364327.xsd
- SapDebmas03E2knkkm00114589673.xsd
- SapDebmas03E2knvam824354065.xsd
- SapDebmas03E2knvdm001293866341.xsd
- SapDebmas03E2knvim824353817.xsd
- SapDebmas03E2knvkh001300182033.xsd
- SapDebmas03E2knvkl824353756.xsd
- SapDebmas03E2knvkm824353755.xsd
- SapDebmas03E2knvlm824353724.xsd
- SapDebmas03E2knvpm002304948594.xsd
- SapDebmas03E2knvvh001310340764.xsd
- SapDebmas03E2knvvl824353415.xsd
- SapDebmas03E2knvvm004310489722.xsd
- SapDebmas03E2vckun000813113929.xsd
- SapIDocControlRecord.xsd

Business object structure

The structure of the ALE outbound business object used in this sample is based on the metadata of the DEBMAS03 business object.

Deploying and configuring for Scenario 1 (ALE outbound)

Scenario 1 of the ALE outbound sample application provides a configured instance of the adapter and all the necessary SCA artifacts, so you are not required to deploy the package and configure the adapter. Simply import the .EAR file to your WebSphere Integration Developer workspace using WebSphere Integration Developer.

After you uncompress the nonemdsample.zip file, perform the following steps to extract the contents of CustomerMasterIDocApp.ear and to import the RAR file.

1. Uncompress the file CustomerMasterIDocApp.ear to a directory of your choice. The CWYAP_SAPAdapter.rar file and various JAR files are now listed in the directory.
2. If the connector project is not already defined in your WebSphere Integration Developer workspace, import the latest CWYAP_SAPAdapter.rar file in WebSphere Integration Developer. Importing the RAR file creates a connector project for the adapter.
3. In WebSphere Integration Developer, create a module and assign it the same name as the EAR file without the letters "App." For example, if the EAR file is CustomerMasterIDocApp.ear, then the module name in this case should be `CustomerMasterIDoc`.
4. Extract the contents of the module JAR file into the module. If prompted, do **not** overwrite the .classpath or .runtime files. For example, extract CustomerMasterIDoc.jar into the module you just created called `CustomerMasterIDoc`.
5. In WebSphere Integration Developer, switch to the Business Integration perspective, right-click on the module, and select **Refresh** from the pop-up menu. You should now see all the .xsd files in the **Data Types** folder.
6. Do the following to associate the module with the SAP connector project:
   a. Right-click the module and select **Properties** from the pop-up menu.
   b. Select **Java build path** from the left-side menu.
   c. Click the Projects tab and select the check box for the connector project you created in 2
7. Do the following to associate the CustomerMasterIDocApp project with the SAP connector project.
   a. In the Java perspective, right-click the CustomerMasterIDocApp project.
   b. Select **Project References** from the left-side menu.
   c. Select the check box for the connector project you created in 2.
   d. Within the CustomerMasterIDocApp project, locate the file application.xml and open it using Deployment Descriptor Editor.
   e. In the **Modules** section, click **Details**.
   f. On the next panel, click on **Add** and add CWYAP_SAPAdapter.rar.
   g. Press **Ctrl**+**S** to save your changes.
8. Select **Project** → **Clean** from the WebSphere Integration Developer menu to rebuild the projects.
9. Start up WebSphere Process Server.
10. Add the project to WebSphere Process Server.

11. Go to the administrative console and ensure that the application has successfully started running.
12. If any processes are not running properly, stop and restart the server.
13. Ensure that the SAP_Auth_Alias has been properly configured with the userID and password required to log on to the SAP application.

The adapter configuration properties are already set to the appropriate values in the adapter instance that is provided with the example. However, you must change the values of the following properties to suit your configuration:

Configuration properties you must set for your configuration

Property	Type	Globalized	Description
ApplicationServerHost	String	Yes	When configuring the adapter to run without load balancing, specifies the IP address or the name of the application server that the adapter logs into. In both cases, the adapter assumes that the name of the gateway host is the same as the value specified for this property.
Password	String	Yes	Password for the adapter's user account on the SAP system.
Username	String	Yes	Name of the adapter's user account on the SAP system.

To change these values, use the WebSphere Process Server administrative console.

Deploying and configuring for Scenario 2 (ALE outbound)

Scenario 2 of the ALE outbound sample application requires you to use the enterprise service discovery wizard to deploy the application package, configure the adapter, and generate the SCA artifacts.

Before you begin to deploy and configure:

- Import the CWYAP_SAPAdapter.RAR file into the project.
- Add the external dependencies (SAP Java Connector (SAP JCo) interface) to the project.

For this task, you must run the Enterprise Service Discovery within IBM WebSphere Integration Developer to set adapter configuration properties. To change the property values later, use the WebSphere Process Server Administrative Console.

1. In the Business Integration Perspective of WebSphere Integration Developer, right-click the frame and from the pop-up menu, select **New → Enterprise Service Discovery**.
2. In the Import Configurations window, select **IBM WebSphere Adapter for SAP Software**.
3. In the Discovery Agent Initialize Properties window, specify the configuration properties to initialize the discovery agent. Be sure to set the **Select Module** field to the value of `ALE`. Properties marked with an asterisk (*) are mandatory.
4. At the bottom of the Discovery Agent Initialize Properties window, click the **Show Advanced** button.

5. When prompted to specify logging options, set the **Logging Level** to `FINEST` to provide the highest level of logging and then click **Next**.
6. In the Metadata Query window, click **Run Query**.
7. Under **Objects Discovered**, drill down to the **Discover By Name** node, and then click the **Filter** button.
8. In the Filter Properties for Discover by Name window, enter `DEBMAS03` (the name of the IDoc in SAP) and click **OK**.
9. Navigate to DEBMAS03 and click the **Add** button.
10. In the Configuration Parameters window, select the **Use Field Name to generate attribute(s)** check box, and click **OK** to add DEBMAS03 to the list of business objects to be imported.
11. Click **Next**.
12. In the Configure Objects window, specify the following:
    - `BODEFS` as the object location
    - A namespace for the business object.
    - `Outbound` as the **Service Type**.
13. Click **Add** to specify the operations.
14. Double-click **EXECUTE** to add it to the list of operations and click **Next** to continue. (The only valid operation for ALE outbound processing is EXECUTE.)
15. In the Generate Artifacts window, specify the module name where the artifacts should be saved, and then click **New** to create a new business integration module.
16. When prompted, enter `CustomerMasterIDoc` as the module name, and click **Finish**.
17. Specify `ScaArtifacts` as the folder name within the module where the Service Description should be saved.
18. Enter `SAP_Auth_Alias` in **J2C Authentication Data Entry** and select the **Deploy connector with module** check box.
19. Select **Use discovered connection properties** and specify value for the connection properties.
20. Click **Finish**. The new CustomerMasterIDoc module is added to the Business Integration Perspective, along with all its artifacts.
21. Export the project to the .EAR file (note that the .EAR file does not yet exist; this step defines it):
    a. Right-click the CustomerMasterIDoc module and select **Export** from the pop-up menu.
    b. Select **EAR file** from the Export window.
    c. Click **Next**.
    d. In the EAR Export window, select `CustomerMasterIDoc` for the .EAR project and specify its destination directory. Then click **Finish**.
22. In the EAR Export window, export the EAR file of the adapter:
    a. Select the .EAR file in the **EAR project** field.
    b. Specify its destination directory in the **Destination** field.
    c. Click **Finish**.
23. Deploy the .EAR file.

The adapter application package, including all its SCA artifacts, is now configured and deployed. You can now run the sample application.

Running the sample application

After you deploy and configure the sample application package and adapter, you can run the application to illustrate how the adapter supports outbound processing of ALE business objects.

1. Start the administrative console and ensure that the ALE outbound sample application has been installed.
2. Ensure that the connection factory properties are correctly configured and that the J2C authentication alias, SAP_Auth_Alias, has the proper credentials to log onto the SAP system.
3. Switch to the Business Integration perspective in WebSphere Integration Developer.
4. To start testing, right-click the name of the module and select **Test → Module**. The WebSphere Business Integration Test Client displays a panel allowing you to select the Configuration, Module, Component, Interface and Operation you wish to test.
5. For the ALE outbound sample application, select the following values:
   - **Configuration**: `Default Module Test`
   - **Module**: `CustomerMasterIDoc`
   - **Component**: `SAPOutboundInterface`
   - **Interface**: `SAPOutboundInterface`
   - **Operation**: `executeSapDebmas03`
6. Populate values of the input business objects.
   a. Enter values that are appropriate for your installation in the following fields of the SapIDocControlRecord business object:
      - NameOfTableStructure
      - Client
      - IdocNumber
      - NameOfBasicType
      - LogicalMessageType
      - SenderPort
      - PartnerTypeOfSender
      - PartnerNumberOfSender
      - ReceiverPort
      - PartnerTypeOfRecipient
      - PartnerNumberOfRecipient

   b. Enter values that are appropriate for your installation in the following fields of the SapDebmas03E2kna1m004 business object, which is a child of the SapDebmas03DataRecord business object.
      - Customeraccountgroup
      - Name1
      - Name2
      - Customernumber
7. Execute the service.
8. Review the application output for confirmation of how the adapter processed the service.

ALE inbound sample application

The ALE inbound sample application illustrates the creation of a customer master IDoc (DEBMAS) in the SAP system. The business object definitions for Scenario 1 (no enterprise service discovery requirement) have been generated specifically for an SAP 4.6B system that uses a DEBMAS03 IDoc type. To run the sample on a system with a newer version of SAP and generate the appropriate business object definitions for your SAP instance, follow the steps in Scenario 2 (enterprise service discovery requirement).

The sample shows how to configure the adapter as an SCA component, and, once deployed, how to create the IDoc. It does not show how to process the IDoc in the SAP system to create a customer master record.

Note: Cloudscape is shipped with WebSphere Process Server and is used as the EDT database for this inbound processing sample. The EDT properties are already set with values for Cloudscape XA connectivity.

Structure of the application package

The sample application files are installed when you install the adapter. They are packaged in an archive file that is installed in the samples folder.

Scenario 1: All-inclusive set of sample files

For Scenario 1 of the ALE inbound sample application, targeted at the Application Integrator, the sample application package includes all the required artifacts, so you do not need to use the enterprise service discovery wizard to obtain these. In a real-world deployment of the adapter, you would generate all these artifacts and configure the adapter using the enterprise service discovery wizard, as illustrated in Scenario 2.

The files for Scenario 1 are archived in the following file: \samples\ nonemdsample.zip.

Uncompress this file to extract the file \ALE\AleCustomerInboundApp.ear to the location of your choice.

The following files are extracted from the .EAR file:

- A configured instance of the adapter that is deployed by default to your local host. The file name is CWYAP_SAPAdapter.rar.
- An SCA module with the following SCA artifacts:
  - AleCustomerInbound.jar
  - AleCustomerInboundEJB.jar
  - AleCustomerInboundEJBClient.jar
  - AleCustomerInboundWeb.war

Scenario 2: Enterprise service discovery wizard requirement

For Scenario 2 of the ALE outbound sample application, targeted at the Data Integrator, the sample application package provides the capability for you to create the SCA artifacts and configure the adapter using the enterprise service discovery wizard. You can, however, access copies of all the files that you eventually generate using the enterprise service discovery wizard, as a way of verifying that what you create with the tool is indeed correct and what the application expects in order to run properly.

All the necessary files, including copies of those you will generate with the enterprise service discovery wizard (for verification after you create your own copies) are archived in the following file: \samples\emdsample.zip.

Uncompress this .zip file to extract the following files for Scenario 2 to the Sample directory:

- SAPIboundInterface.export
- SAPInboundInterface.wsdl
- sca.module
- SapDebmas03.xsd
- SapDebmas03BG.xsd
- SapDebmas03DataRecord.xsd
- SapDebmas03E2kna11001855049811.xsd
- SapDebmas03E2kna1h001214229578.xsd
- SapDebmas03E2kna1l851889735.xsd
- SapDebmas03E2kna1m0041875542811.xsd
- SapDebmas03E2knasm611830969.xsd
- SapDebmas03E2knb1h0011787596074.xsd
- SapDebmas03E2knb1l722836684.xsd
- SapDebmas03E2knb1m005498689821.xsd
- SapDebmas03E2knb5m482777918.xsd
- SapDebmas03E2knbkm00265919925.xsd
- SapDebmas03E2knexm732594868.xsd
- SapDebmas03E2knkam112849634.xsd
- SapDebmas03E2knkkh0011769946559.xsd
- SapDebmas03E2knkkl705983448.xsd
- SapDebmas03E2knkkm0011932217898.xsd
- SapDebmas03E2knvam1306131080.xsd
- SapDebmas03E2knvdm001288220673.xsd
- SapDebmas03E2knvim184600160.xsd
- SapDebmas03E2knvkh0011239217145.xsd
- SapDebmas03E2knvkl712977085.xsd
- SapDebmas03E2knvkm516463579.xsd
- SapDebmas03E2knvlm1280487874.xsd
- SapDebmas03E2knvpm0021655001224.xsd
- SapDebmas03E2knvvh001709072932.xsd
- SapDebmas03E2knvvl1873621688.xsd
- SapDebmas03E2knvvm0041561930042.xsd
- SapDebmas03E2vckun00012062206.xsd
- SapIDocControlRecord.xsd

Business object structure

The business object structure of the ALE inbound business object used in this sample is based on the metadata of the DEBMAS03 business object.

Deploying and configuring for Scenario 1 (ALE inbound)

Scenario 1 of the ALE inbound sample application provides a configured instance of the adapter and all the necessary SCA artifacts, so you are not required to

deploy the package and configure the adapter. Simply import the .EAR file using WebSphere Integration Developer to your WebSphere Integration Developer workspace.

After you uncompress the nonemdsample.zip file, perform the following steps to extract the contents of AleCustomerInboundApp.ear and to import the RAR file.

1. Uncompress the file AleCustomerInboundApp.ear to a directory of your choice. The CWYAP_SAPAdapter.rar file and various JAR files are now listed in the directory.
2. If the connector project is not already defined in your WebSphere Integration Developer workspace, import the latest CWYAP_SAPAdapter.rar file in WebSphere Integration Developer. Importing the RAR file creates a connector project for the adapter.
3. In WebSphere Integration Developer, create a module and assign it the same name as the EAR file without the letters "App." For example, if the EAR file is AleCustomerInboundApp.ear, then the module name in this case should be `AleCustomerInbound`.
4. Extract the contents of the module JAR file into the module. If prompted, do **not** overwrite the .classpath or .runtime files. For example, extract AleCustomerInbound.jar into the module you just created called `AleCustomerInbound`.
5. In WebSphere Integration Developer, switch to the Business Integration perspective, right-click on the module, and select **Refresh** from the pop-up menu. You should now see all the .xsd files in the **Data Types** folder.
6. Do the following to associate the module with the SAP connector project:
   a. Right-click the module and select **Properties** from the pop-up menu.
   b. Select **Java build path** from the left-side menu.
   c. Click the Projects tab and select the check box for the connector project you created in 2
7. Do the following to associate the AleCustomerInboundApp project with the SAP connector project.
   a. In the Java perspective, right-click the AleCustomerInboundApp project.
   b. Select **Project References** from the left-side menu.
   c. Select the check box for the connector project you created in 2.
   d. Within the AleCustomerInboundApp project, locate the file application.xml and open it using Deployment Descriptor Editor.
   e. In the **Modules** section, click **Details**.
   f. On the next panel, click on **Add** and add CWYAP_SAPAdapter.rar.
   g. Press **Ctrl**+**S** to save your changes.
8. Select **Project** → **Clean** from the WebSphere Integration Developer menu to rebuild the projects.
9. Start up WebSphere Process Server.
10. Add the project to WebSphere Process Server.
11. Go to the administrative console and ensure that the application has successfully started running.
12. If any processes are not running properly, stop and restart the server.
13. Ensure that the SAP_Auth_Alias has been properly configured with the userID and password required to log on to the SAP application.
14. Using Business Object Editor, manually edit the file SapDebmas03.xsd in the **Data Types** folder as follows: replace the `<sapasi:MsgType/>` tag with

<sapasi:MsgType>DEBMAS</sapasi:MsgType>. For details about editing business objects using the Business Object Editor, refer to the WebSphere Integration Developer documentation.

The adapter configuration properties are already set to the appropriate values in the adapter instance that is provided with the example. However, you must change the values of the following properties to suit your configuration:

Configuration properties to set for your configuration

Property	Type	Globalized	Description
ApplicationServerHost	String	Yes	When configuring the adapter to run without load balancing, specifies the IP address or the name of the application server that the adapter logs into. In both cases, the adapter assumes that the name of the gateway host is the same as the value specified for this property.
Password	String	Yes	Password for the adapter's user account on the SAP system.
Username	String	Yes	Name of the adapter's user account on the SAP system.

To change these values, use the WebSphere Process Server Administrative Console.

Deploying and configuring for Scenario 2 (ALE inbound)

Scenario 2 of the of the ALE inbound sample application requires you to use the Enterprise Service Discovery to deploy the application package, configure the adapter, and generate the SCA artifacts.

Before you begin to deploy and configure:

- Import the CWYAP_SAPAdapter.RAR file into the project.
- Add the external dependencies (SAP Java Connector (SAP JCo) interface) to the project.

For this task, you must run the Enterprise Service Discovery within WebSphere Integration Developer to set adapter configuration properties. To change the property values later, use the WebSphere Process Server Administrative Console.

1. In the Business Integration Perspective of WebSphere Integration Developer, right-click the frame and from the pop-up menu, select **New → Enterprise Service Discovery**.
2. In the Import Configurations window, select **IBM WebSphere Adapter for SAP Software**.
3. In the Discovery Agent Initialize Properties window, specify the configuration properties to initialize the discovery agent. Be sure to set the **Select Module** field to the value of `ALE`. Properties marked with an asterisk (*) are mandatory.
4. At the bottom of the Discovery Agent Initialize Properties window, click the **Show Advanced** button.
5. When prompted to specify logging options, set the **Logging Level** to `FINEST` to provide the highest level of logging and then click **Next**.
6. In the Metadata Query window, click **Run Query**.

7. Under **Objects Discovered**, drill down to the **Discover By Name** node, and then click the **Filter** button.
8. In the Filter Properties for Discover by Name window, enter `DEBMAS03` (the name of the IDoc in SAP) and click **OK**.
9. Navigate to DEBMAS03 and click the **Add** button.
10. In the Configuration Parameters window, select the **Use SAP Field Name to generate attribute(s)** check box, and click **OK** to add DEBMAS03 to the list of business objects to be imported.
11. Click **Next**.
12. In the Configure Objects window, specify the following properties to configure objects that will be imported:
    - The object location. In this example, enter `BoDefs` to indicate that all business object schemas will be placed in the BoDefs directory.
    - A namespace for the business object.
    - `Inbound` for the **Service Type**,
13. Click **Add** to specify the operations.
14. From the list of available operations (**Create**, **Delete**, and **UpdateWithDelete**), double-click **Create** to add it to the list of operations.
15. Click **OK** . This adds Create to the list of operations in the Configure Objects window.
16. In the Generate Artifacts window, specify the module name where the artifacts should be saved, and then click **New** to create a new business integration module.
17. In the Generate Artifacts window, when prompted, enter `AleCustomerInbound` as the module name, and click **Finish**.
18. In the Generate Artifacts window, specify a folder name within the module where the Service Description should be saved and select the **Deploy connector with module** check box.
19. Select **Use discovered connection properties**.
20. In **J2C Authentication Data Entry**, enter `SAP_Auth_Alias`.
21. Scroll down to enter values that are appropriate for your installation in the following inbound connection properties:
    - GatewayHost
    - GatewayService
    - RfcProgramID
    - Client
    - NumberOfUsers
    - UserName
    - Password
    - Language
    - ApplicationServerHost
    - SystemNumber
    - EDTTableName
    - EDTDriverName
    - EDTDatabaseName
    - EDTUserName
    - EDTUserPassword
    - EDTSchemaName

22. Click **Finish**.
23. Using Business Object Editor, manually edit the file SapDebmas03.xsd in the **Data Types** folder as follows: replace the `<sapasi:MsgType/>` tag with `<sapasi:MsgType>DEBMAS</sapasi:MsgType>`. For details about editing business objects using the Business Object Editor, refer to the WebSphere Integration Developer documentation.

The business integration module project with its SCA artifacts and corresponding business objects has been created. The next task is to generate reference bindings for the module.

Generating reference bindings for Scenario 2:

This topic describes how to generate reference bindings for the ALE inbound sample application.

1. In the Business Integration Perspective of WebSphere Integration Developer, right-click the ALECustomerInbound SCA module, and select **Open With → Assembly Editor**. The Assembly Diagram window appears with the module's Export component in view.
2. To create a new component, click the top-most icon in the left-side (vertical) frame of the Assembly Diagram Window. A new menu of icons appears.
3. Click the top-most icon in the new menu of icons (it has hover-help that reads "Component (with no implementation type)"). The cursor changes to the placement icon.
4. Click the palette to add the new component to the Assembly Diagram window.
5. Click and drag the module's Export component to the new component. This draws a wire from the Export component to the new component and displays the Add Wire window.
6. In the Add Wire window, click **OK**.
7. Right-click the new component and select **Generate Implementation → Java**. This creates a Java component that will act as an endpoint.
8. In the Generate Implementation window, select the package in which the Java code will be created and click **OK**. A Java file in an editor window appears.
9. Edit the Java file. For example, you may wish to write code to print trace and log messages.
10. Save the Java file.

The reference bindings across components in the module are generated. Now you can export the project to the sample application .EAR file.

Exporting and deploying the project for Scenario 2:

After you create reference bindings across the components in the sample module, you are ready to export the project to the sample application .EAR file and then deploy the .EAR file.

The following steps are performed in the Business Integration perspective of WebSphere Integration Developer.

1. Perform the following steps to export the project to the .EAR file (note that the .EAR file does not yet exist; this step defines it):
   a. Right-click the `AleCustomerInbound` module and select **Export** from the pop-up menu.

b. Select **EAR file** from the Export window.
c. Click **Next**.
d. In the EAR Export window, select `AleCustomerInboundApp` for the .EAR project and specify its destination directory. Then click **Finish**.

2. Deploy the .EAR file.

The adapter application package, including all its SCA artifacts, is configured and deployed. You can now run the sample application.

Running the sample application

After you deploy and configure the sample application package and adapter, you can run the application to illustrate how the adapter supports inbound processing of ALE objects.

1. Start the administrative console and ensure that the ALE inbound sample application has been installed.
2. Ensure that the J2C activation specification properties are configured correctly and that the J2C authentication alias, SAP_Auth_Alias, has the proper credentials to log onto the SAP instance.
3. Switch to the Business Integration perspective in WebSphere Integration Developer.
4. To start testing, right-click the name of the module and select **Test → Attach**. The test client displays the Events window.
5. Examine the Configurations window and confirm that a monitor exists for the export.
6. Return to the **Events** window, then click the **Continue** button. The Select Deployment Location window appears.
7. Select the WebSphere Process Server instance you want to test and click the **Finish** button. The Starting the integration test client window appears.
8. Use the WE19 transaction in the SAP client user interface to send an ALE IDoc from the SAP instance.
   - This creates an event for the ALE inbound application.
   - After the adapter has successfully processed the event, the Request parameters window is populated with the data object returned by the adapter.

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
577 Airport Blvd., Suite 800
Burlingame, CA 94010
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, a nd 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.

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

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both:

i5/OS
IBM
the IBM logo
AIX
AIX 5L
CICS
CrossWorlds
DB2
DB2 Universal Database
Domino
HelpNow
IMS
Informix
iSeries
Lotus
Lotus Notes
MQIntegrator
MQSeries
MVS
Notes
OS/400
Passport Advantage
pSeries
Redbooks
SupportPac
WebSphere
z/OS

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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/).

WebSphere Adapters, Version 6.0

Printed in USA