WebSphere Adapters

Adapter for JDBC User Guide

Version 6.0

Note
Before using this information, be sure to read the general information in "Notices" on page 83.

7April2006

This edition of this document applies to IBM WebSphere Adapter for JDBC (5724L77), Version 6.0, 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 JDBC Version 6.0 User Guide

The IBM(R) WebSphere(R) Adapter for JDBC provides bidirectional connectivity between J2EE applications and enterprise information systems.

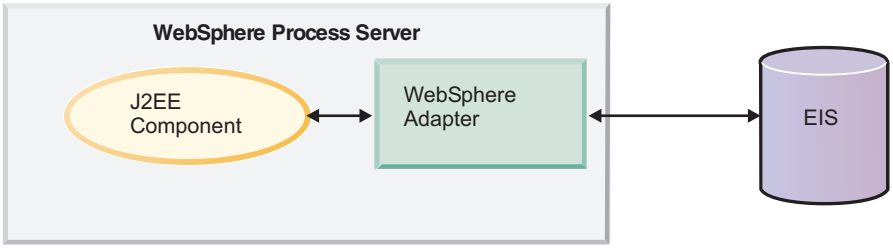
Product overview

This topic introduces basic concepts about the WebSphere(R) Adapter for JDBC, its architecture and its environmental requirements.

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:

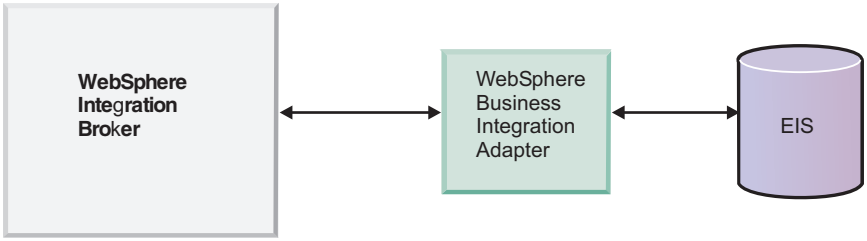
- An implementation of the (J2EE) Connector Architecture (JCA), version 1.5 that supports WebSphere Process Server
- An enterprise metadata discovery component— you use this component with the enterprise service discovery wizard to introspect the EIS— to generate business objects and other service component 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

Related concepts

"Sample: Updating a database application" on page 58
Sample files are provided with the IBM(R) WebSphere(R) Adapter for JDBC so that you can use them to practice updating specific tables in a database application. Two step-by-step scenarios are provided; one is targeted to the application integrator and the other is for the data integrator. Depending on your role, you can practice generating business objects, deploying the adapter, and configuring the adapter to communicate between J2EE(TM) applications and enterprise information systems.

Task roadmap: IBM WebSphere Adapter for JDBC

The IBM(R) WebSphere(R) Adapter for JDBC facilitates the exchange of business objects at the database level between J2EE(TM) applications and database provider enterprise information systems (EISs).

Task	Description
Installing the adapter	Information describing installation considerations specific to the Adapter for JDBC, with a link to the installation instructions.
Creating the adapter project	You will create an adapter project, then generate business objects and service constructs, set configuration properties and, finally, deploy the adapter project to the application server.
Deploying the adapter project	You will export the project to an enterprise application archive (EAR) file and install this file on the application server.
Configuring the adapter on the server	You can reconfigure properties that the adapter uses to set up a communication channel to a specific database application.
Troubleshooting	You will set up a log file to track the status of event processing, and determine the level of errors or warnings to be captured in the log file. You will learn how to use IBM Software Support assistance and how to access technotes.

Task	Description
Updating a sample database application	You will use sample files provided with the Adapter for JDBC to practice deploying, configuring, and running a sample database application. Two scenarios are provided, offering different levels of complexity. The second scenario includes generating business objects.

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 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

Architecture

The WebSphere(R) Adapter for JDBC is a resource adapter that provides bidirectional connectivity between J2EE(TM) applications and enterprise information systems (EISs). For such applications, the exchange of data, which is in the form of business objects, happens at the database level.

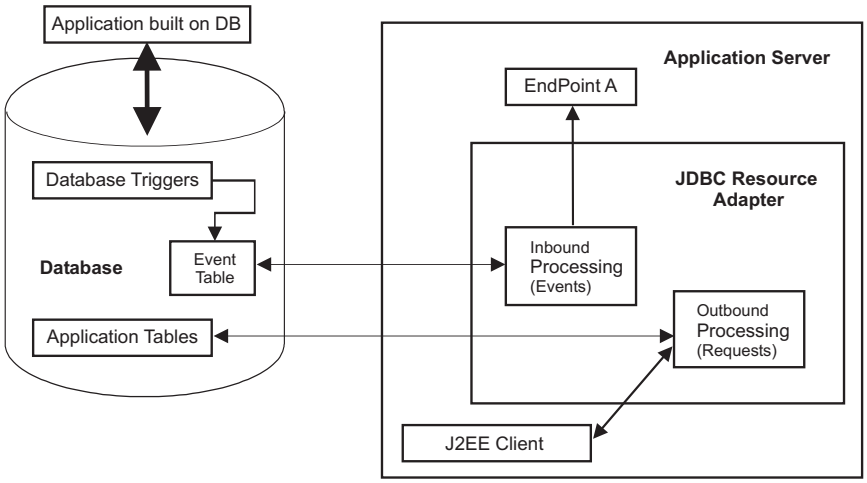
The adapter for JDBC communicates with an EIS which is a database provider. Updates to a database might need to be applied to another enterprise information system, and changes to data in an EIS might need to be applied to a database. The resource adapter can integrate with any application built on a database having a JDBC driver that supports the JDBC 2.0 or later specification. Examples include IBM(R) DB2(R), Oracle, Microsoft(TM) SQLServer, Sybase, and Informix databases.

To support integration, the resource adapter processes requests received from any EIS and processes events generated as a result of database updates. The adapter transmits these events to various predefined endpoints in the application server. *Endpoints* are J2EE applications or other client consumers of the event. Data updates that are made to tables in a database can be automatically propagated to other applications connected to the application server, such as Siebel, PeopleSoft, and Oracle applications, through event notifications posted in an event store. The adapter updates the database tables using SQL queries or stored procedures, as specified in the business objects.

The Adapter for JDBC supports integration of databases accessible through the JDBC application programming interface (API) by providing inbound and outbound support under the Java(TM) 2 Enterprise Edition (J2EE) Connector Architecture (JCA). Under *outbound* operations, a business object is passed to the adapter as a request that is processed according to the operation specified in the business object, either create, retrieve, update, delete, or retrieveall. Requests are received from different EIS applications that need to have the updates applied to the database being managed by the adapter. Processing of these requests results in the creation, retrieval, update or deletion of rows in the corresponding database tables.

Under *inbound* operations, as data is changed in the application tables in the database, appropriate events are inserted into an event store, along with relevant information such as key values. To capture the changed data, triggers are placed on the respective tables, or other methods are used, such as Oracle Change Data Capture, which is provided for Oracle databases. The Adapter for JDBC polls the event store and retrieves a batch of events. These events are processed, and each event is used to construct a JDBC business graph. The business graph is then dispatched to the endpoints that have a subscription for the specific business object. Only after-image support is provided for inbound operations.

The figure titled "Processing within the JDBC adapter" shows inbound and outbound operations.



Processing within the JDBC adapter

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.

Business objects overview

This set of topics describes the structure, attribute properties, supported operations, application-specific information and naming conventions for business objects. It contains information about transaction management.

Business object structure

Each business object corresponds to a database table or view, and each simple attribute within the object corresponds to a column in that table or view.

A *simple attribute* is an attribute that represents a single value, such as a String, Integer, or Date. Thus, attributes within the same business object cannot be stored in different database tables; however, the following situations are possible:

- The database table might have more columns than the corresponding business object has simple attributes; that is, some columns in the database are not represented in the business object. Only those columns needed for processing of the business object should be included in your design.
- The business object might have more simple attributes than the corresponding database table has columns; that is, some attributes in the business object are not represented in the database. The attributes that do not have a representation in the database either have no application-specific information, are set with default values, or specify stored procedures.
- The business object can represent a view that spans multiple database tables. The adapter can use such a business object when processing events triggered in the application, such as Create, Update, and Delete. When processing business object requests, however, the adapter can use such a business object only for Retrieve and RetrieveAll requests.

Note: If a business object is based on a stored procedure (SP), each simple attribute (other than the special SP attributes) may or may not have application-specific information. For more information see "Stored procedure definition."

Business objects can be either flat or hierarchical. All of the attributes of a flat business object are simple and represent one row in the database table. The term *hierarchical* business object refers to a complete business object, including all the child business objects that it contains at any level. The term *individual* business object refers to one business object, independent of child business objects that it might contain or that contain it. The individual business object can represent a view that spans multiple database tables. The term *top-level* business object refers to the individual business object at the top of the hierarchy that does not itself have a parent business object.

A hierarchical business object has attributes that represent a child business object, an array of child business objects, or a combination of the two. In turn, each child business object can contain a child business object or an array of business objects, and so on. A *single-cardinality relationship* occurs when an attribute in a parent business object represents one child business object. In this case, the attribute is of the same type as the child business object.

A *multiple-cardinality relationship* occurs when an attribute in the parent business object represents an array of child business objects. In this case, the attribute is of the same type as the child business objects.

The adapter supports the following relationships among business objects:

- Single-cardinality relationships
- Single-cardinality relationships and data without ownership
- Multiple-cardinality relationships

In each type of cardinality, the relationship between the parent and child business objects is described by the application-specific information of the key attribute in the business object storing the relationship. For more information on this application-specific information, see "ForeignKey."

Related reference

"Stored procedure definition" on page 25
Stored procedures are defined at the verb level. Each stored procedure definition consists of the following elements: StoredProcedureType, StoredProcedureName, ResultSet, and Parameters.

"Business object attribute properties" on page 11
Business object architecture defines various properties that apply to attributes. This section describes how the adapter interprets these properties.

Single-cardinality relationships in business objects

This topic describes the single-cardinality relationships that this adapter supports.

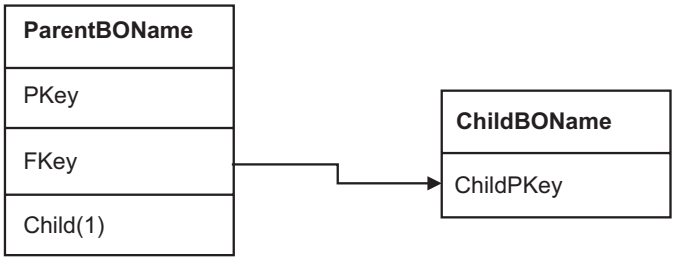
Single-cardinality relationships

Typically, a business object that contains a single-cardinality child business object has at least two attributes that represent the relationship. The type of one attribute is the same as the child's type. The other attribute is a simple attribute that contains the child's primary key as a foreign key in the parent. The parent has as many foreign key attributes as the child has primary key attributes.

Because the foreign keys that establish the relationship are stored in the parent, each parent can contain only one child business object of a given type.

The figure titled "Typical single-cardinality relationship" illustrates a typical single-cardinality relationship. In the example, FKey in the ParentBOName box is

the simple attribute that contains the child's primary key, and Child(1), also in the ParentBOName box, is the attribute that represents the child business object.



Typical single-cardinality relationship

A parent business object can have a single-cardinality OWNERSHIP child and a single-cardinality NOOWNERSHIP child.

Single-cardinality relationships and data without ownership

Typically, each parent business object owns the data within the child business object that it contains. For example, if each Customer business object contains one Address business object, when a new customer is created, a new row is inserted into both the customer and address tables. The new address is unique to the new customer. Likewise, when deleting a customer from the customer table, the customer's address is also deleted from the address table.

However, situations can occur in which multiple hierarchical business objects contain the same data, which none of them owns. For example, assume that an Address business object has a *StateProvince[1]* attribute that represents the *StateProvince lookup table with single cardinality*. Because the lookup table is rarely updated and is maintained independently of the address data, creation or modification of address data does not affect the data in the lookup table. The adapter either finds an existing state or province name or fails. It does not add or change values in the lookup table.

When multiple business objects contain the same single-cardinality child business object, the foreign key attribute in each parent business object must specify the relationship as `NOOWNERSHIP`. When an application server sends the adapter a hierarchical business object with a Create, Delete, or Update request, the adapter ignores single-cardinality children contained without ownership. The adapter performs only retrieve operations on these business objects. If the adapter fails to retrieve such a single-cardinality business object, it returns an error and stops processing.

Denormalized data and data without ownership

In addition to facilitating the use of static lookup tables, containment without ownership provides another capability: synchronizing normalized and denormalized data.

Synchronization of normalized to denormalized data: Specifying a relationship as `NOOWNERSHIP` enables you to create or change data when you synchronize from a normalized application to a denormalized one. For example, assume that a normalized source application stores data in two tables, A and B. Assume further that the denormalized destination application stores all the data in one table such that each entity A redundantly stores B data.

In this example, to synchronize a change in table B data from the source application to the destination application, you must trigger a table A event whenever table B data changes. In addition, because table B data is stored redundantly in table A, you must send a business object for each row in table A that contains the changed data from table B.

Note: When making updates to denormalized tables, ensure that each record has a unique key so that multiple rows are not modified as a result of one update. If such a key does not exist, the Adapter for JDBC provides an error stating that multiple records have been updated.

Synchronization of denormalized to normalized data: When synchronizing data from a denormalized source application to a normalized destination application, the adapter does not create, delete, or update data contained without ownership in the normalized application.

When synchronizing data to a normalized application, the adapter ignores all single-cardinality children contained without ownership. To create, remove, or modify such child data, you must process the data manually.

Multiple-cardinality relationships in business objects

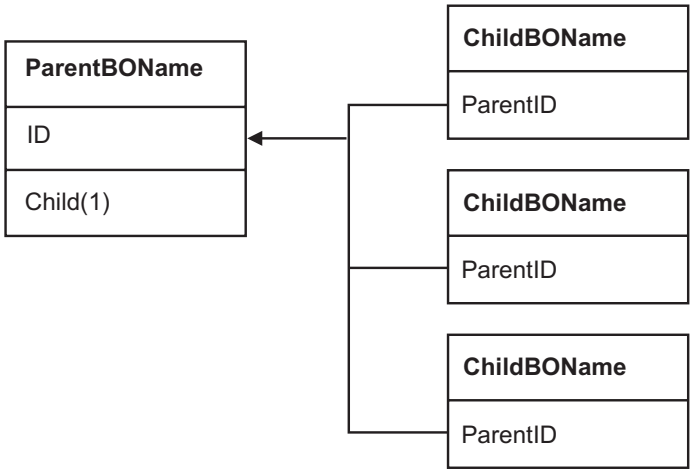
The adapter supports multiple-cardinality relationships.

Typically, a business object that contains an array of child business objects has only one attribute that represents the relationship. The type of the attribute is an array of the same type as the child business objects. For a parent to contain more than one child, the foreign keys that establish the relationship are stored in the child.

Therefore, each child has at least one simple attribute that contains the parent's primary key as a foreign key. The child has as many foreign key attributes as the parent has primary key attributes.

Because the foreign keys that establish the relationship are stored in the child, each parent can have zero or more children.

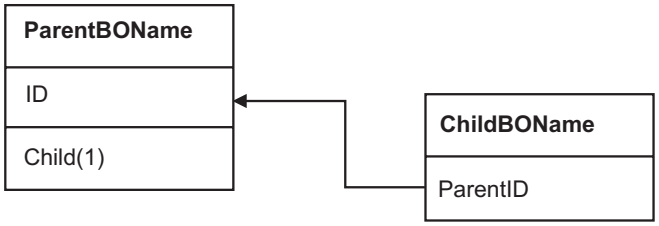
The figure titled "Multiple-cardinality business object relationship" illustrates a multiple-cardinality relationship. In the example, ParentId in the three ChildBOName boxes is the simple attribute that contains the parent's primary key, and Child1 in the ParentBOName box is that attribute that represents the array of child business objects.



Multiple cardinality business object relationship

A multiple cardinality relationship could be an N=1 relationship. Some applications store one child entity so that the parent-child relationship is stored in the child rather than in the parent. In other words, the child contains a foreign key whose value is identical to the value stored in the parent's primary key.

Applications use this type of relationship when child data does not exist independently of its parent and can be accessed only through its parent. Such child data requires that the parent and its primary key value exist before the child and its foreign key value can be created. The figure titled "Multiple cardinality relationship with N=1" shows this type of relationship.



Multiple cardinality relationship with N=1

Business object attribute properties

Business object architecture defines various properties that apply to attributes. This section describes how the adapter interprets these properties.

The following table titled "Attribute properties" gives the interpretation and settings for these properties.

Attribute properties

Properties	Interpretation and settings
Cardinality	Each business object attribute that represents a child or an array of child business objects has the value of single (1) or multiple (n) cardinality, respectively. All attributes that represent child business objects also have a ContainedObjectVersion property (which specifies the child's version number) and a Relationship property (which specifies the value Containment).
Foreign Key	When arrays of child business objects whose cardinality is *n* are retrieved, foreign keys are used in the WHERE clause of SELECT statements. **Note:** The adapter does not support specifying an attribute that represents a child business object as a foreign key. The RetrieveAll verb overrides the use of keys and foreign keys.

Properties	Interpretation and settings
Key	At least one simple attribute in each business object must be specified as the key. **Note:** The adapter does not support specifying an attribute that represents a child business object or an array of child business objects as a key attribute. If the key property is set to `true` for a simple attribute, the adapter adds that attribute to the WHERE clause of the SELECT statement and UPDATE SQL statements that it generates while processing the business object. The RetrieveAll verb overrides the use of keys and foreign keys.
Name	This property represents the unique name of the attribute, if it is a simple attribute, or the name of the business object, if it is a child business object.
Required	Specifies whether an attribute must contain a value. If this property is set to `true` for a container whose cardinality is single (1), then the adapter requires that the parent business object contain a child business object for this attribute. Business objects that are passed to the adapter for create, update, and delete operations must also contain a child business object. Cardinality is single (1) for simple attributes and multiple (n) for container attributes. The adapter causes a create operation to fail if a business object does not have a valid value or a default value for a required attribute. It also fails if no data is available upon retrieval from the database for this object.
Type	The type of the attribute (such as Integer, String, Date, Boolean, Double, or Float) if it is a simple attribute, or the type of business object if it is a child business object. When the adapter encounters an attribute of a type that it does not support, the adapter wraps the value in quotation marks and handles the value as character data.

Related concepts

"Business object structure" on page 7
Each business object corresponds to a database table or view, and each simple attribute within the object corresponds to a column in that table or view.

Supported operations

The adapter performs inbound and outbound operations. After-image and delta support are provided for outbound operations. The adapter provides only after-image support for inbound operations. The supported operations are listed here.

Outbound operations

The adapter supports two business object styles that relate to the amount and purpose of information conveyed by the business object: after-image and delta. An *after-image* is the state of a business object after all changes have been made to it. A *delta* is a business object used in an update operation that contains only key values and the data to be changed. The adapter provides both after-image and delta support for the following operations:

- Create
- Update
- Delete
- ApplyChanges
- Retrieve
- RetrieveAll

Verbs are definable only for after-image business objects. A *verb* reflects the state of the business object, whereas an *operation* reflects the operation to be performed by the adapter. The top-level verbs supported for after-image are:

- Create
- Update
- Delete
- UpdateWithDelete

Inbound operations

The adapter provides only after-image support for inbound operations. When the adapter receives a business object for updating, it assumes that the business object represents the desired state of the data after the update. These are the supported operations:

- Create
- Update
- Delete

Transaction management

The Adapter for JDBC supports both local and XA transactions.

In the adapter, a transaction is an isolated interaction with the back-end database, or enterprise information system (EIS). A transaction can consist of multiple operations on the database that are performed as an atomic unit. These operations are not affected by simultaneously occurring operations from other client applications of the database.

The Adapter for JDBC supports transactions only if the back-end database supports transactions. The types of transactions that are supported are local and XA transactions:

- In a *local transaction*, a given client application defines the start and end of the transaction with the database. It uses a one-phase-commit protocol.
- In an *XA transaction*, the transaction spans multiple heterogeneous databases. It uses global or two-phase-commit protocol.

The adapter supports XA transactions for IBM(R) DB2(R) and Oracle databases.

Use the properties XADataSourceName and XADatabaseName with XA transactions. See the "J2C connection factory properties" in the "Reference" section for details about these properties.

Note: If you are using a DB2 database, you must configure the XADatabaseName property.

The "Reference" section has example values for properties used with local and XA transactions.

Related reference

"J2C connection factory properties" on page 72
J2C connection factory properties are used to configure a target enterprise information system (EIS) instance. These properties affect outbound processing and correspond to the ManagedConnectionFactory interface of the J2EE(TM) Connector Architecture Specification.

"Reference" on page 68
The section describes the properties for configuration, connection, and object selection.

Inbound support

The Adapter for JDBC supports inbound event management with asynchronous event delivery.

Asynchronous event delivery is accomplished through use of an event table, called an event store, and a staging table, or event distribution table. For any changes in the user tables, the application populates the event table in the enterprise information system (EIS). Updates are made by placing triggers on the user tables that record events in the event table corresponding to the updates to the user table.

The staging table is deployed as part of an XA-compliant database, such as Cloudscape(TM), which is included with WebSphere(R) Application Server, Version 6.0. When an event is retrieved from the EIS, a reference to the event is written to the staging table for each active endpoint. Each event in the staging table is delivered to its corresponding endpoint as part of a unique XA transaction controlled by the application server. When the endpoint consumes its event and the transaction is committed, the reference to the event in the staging table is removed for that and only that endpoint.

The event table is described in the following table.

Event table

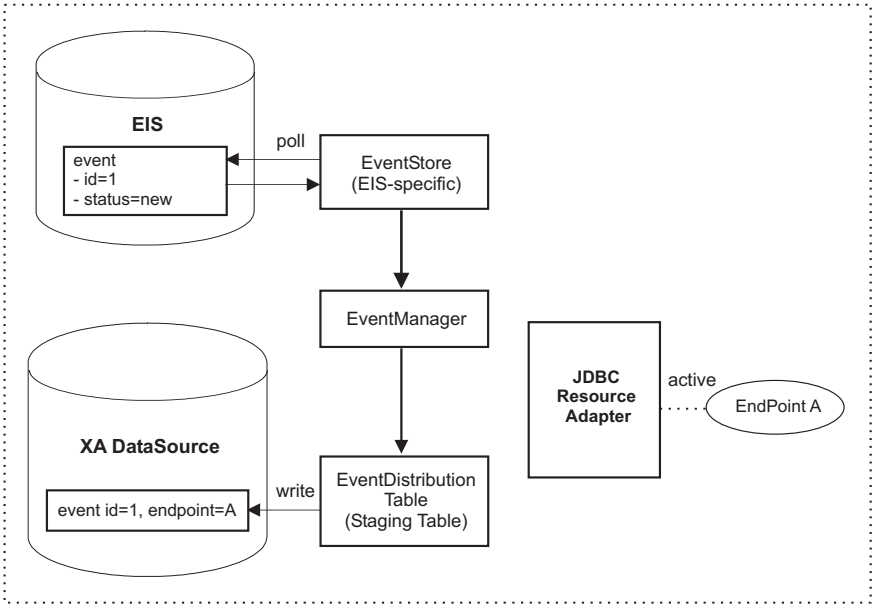
Name	Type	Description
event_id	Number	Unique Eventid that is a Primary key for the table
object_key	String	Delimited string that contains keys for the business object (not null)
object_name	String	Name of the business object (not null)
object_function	String	Operation corresponding to the event (Delete, Create, Update, and so on (not null)
event_priority	Number	Not null
event_time	Date	Date and time when event was generated
event_status	Number	In Progress, Success, Failure, and so on (not null)

Name	Type	Description
event_comment	String	Description

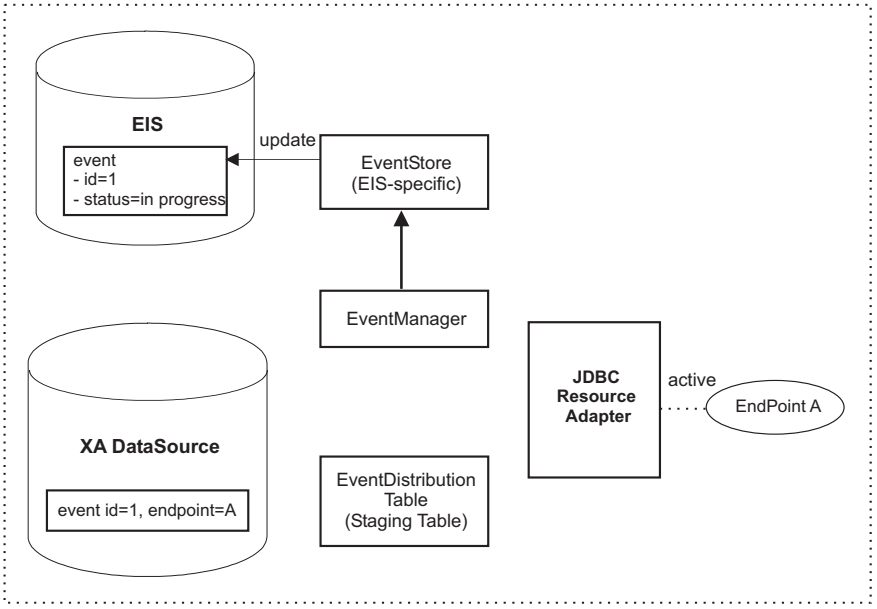
Event processing in the Adapter for JDBC

The complete flow of event processing by the Adapter for JDBC is depicted in four steps that are shown in the following four figures.

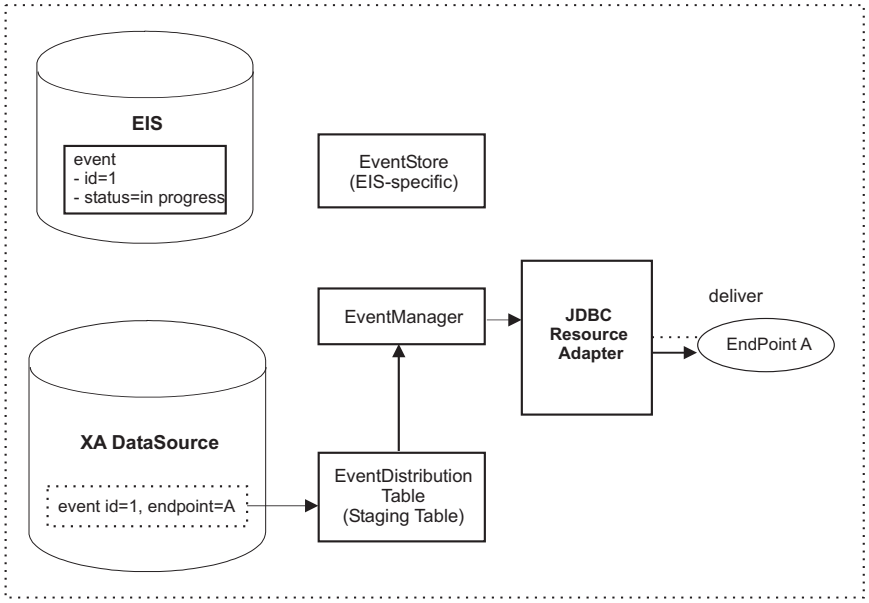
The first figure shows the first step, in which the event has been detected in the EIS and the record has been added to the staging table for each active endpoint.



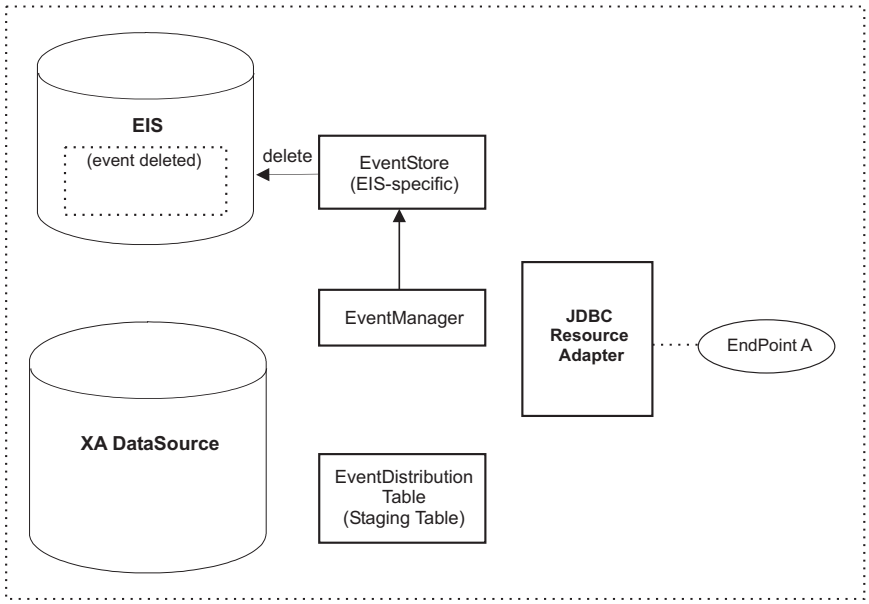
The next figure shows the second step, in which the event in the EIS is marked in-progress.



The next figure shows the third step, in which the event is published to all active endpoints, by using the staging table as a guide for which endpoints require delivery. Simultaneously, the event is deleted from the staging table as part of individual XA transactions encompassing each endpoint and its specific record in the staging table. In effect, each endpoint is isolated from the others in terms of its receipt and processing of the event.



In the fourth step, once all endpoints have received the event, as determined by a lack of remaining references in the staging table, the original event is deleted from the EIS. Steps 1 through 4 repeat for every event.



Business object processing for outbound operations

This set of topics outlines the steps the adapter takes when creating, retrieving, updating, or deleting a business object. The adapter processes hierarchical business

objects recursively; that is, it performs the same steps for each child business object until it has processed all individual business objects.

Business object comparison

At various points in the business object verb operations, the adapter compares two business objects to see if they are the same. For example, during an update operation, the adapter determines whether a particular business object exists in an array of business objects. To perform the check, the adapter compares the business object to each business object within the array. For two business objects to be identical, the following two conditions must be satisfied:

- The type of the business objects being compared must be the same. For example, a Customer business object is never considered identical to a Contact business object, even if all of their attributes are exactly the same.
- All corresponding key attributes in the two business objects must contain identical values. If a key attribute is not set in both business objects, the adapter considers them identical, However, if a key attribute is set in one business object, but not in the other, the business objects are not identical.

Create operation

When given a hierarchical business object, the create operation recursively traverses the business object, creating rows corresponding to each table.

Here are the details:

1. The create operation recursively inserts each single-cardinality child business object contained with ownership into the database. In other words, the adapter creates the child and all child business objects that the child and its children contain.

   If the business object definition specifies that an attribute represents a child business object with single cardinality and that attribute is empty, the adapter ignores the attribute. However, if the business object definition requires that the attribute represent a child, and it does not, the adapter returns an error and stops processing.
2. The create operation retrieves and checks the existence of each single-cardinality child business object contained without ownership. If the retrieve operation is unsuccessful, indicating that the child does not exist in the database, the adapter returns an error and stops processing. If the retrieve operation is successful, the adapter recursively updates the child business object.

   Note: For this approach to work correctly when the child business object exists in the application database, primary key attributes in child business objects must be cross-referenced correctly on create operations. If the child business object does not exist in the application database, the primary key attributes must not be set.
3. It inserts the top-level business object in the database as follows:
   a. It sets each of its foreign key values to the primary key values of the corresponding child business object represented with single cardinality. Because values in child business objects can be set by database sequences or counters or by the database itself during the creation of the child, this step ensures that the foreign key values in the parent are correct before the adapter inserts the parent in the database.
   b. It generates a new, unique ID value for each attribute that is set automatically by the database. The name of the database sequence or

counter is stored in the attribute's application-specific information. If an attribute has an associated database sequence or counter, the value generated by the adapter overwrites any value passed in by the application server. For more information on specifying a database sequence or counter, see UID=AUTO in "Application-specific information for simple attributes."

c. It inserts the top-level business object into the database.

4. It processes each of its multiple-cardinality child business objects as follows:
   a. It sets the foreign key values in each child to reference the value in the corresponding primary key attributes in the parent. Because the parent's primary key values might have been generated during the creation of the parent, this ensures that the foreign-key values in each child are correct before the adapter inserts the child into the database.
   b. It inserts each of its multiple-cardinality child business objects into the database.

Retrieve operation

This topic describes the steps the adapter takes to retrieve a hierarchical business object.

The adapter performs the retrieve operation as follows:

1. It removes all child business objects from the top-level business object received. In other words, it makes a copy of the top-level business object without any children.
2. It retrieves the top-level business object from the database.
   - If the retrieval returns one row, the adapter continues processing.
   - If the retrieval returns no rows, indicating that the top-level business object does not exist in the database, the adapter returns the error `RecordNotFoundException`.
   - If the retrieval returns more than one row, the adapter returns an error.
3. It recursively retrieves all multiple-cardinality child business objects.

   Note: The adapter does not enforce uniqueness when populating an array of business objects. It is the database's responsibility to ensure uniqueness. If the database returns duplicate child business objects, the adapter returns duplicate children.
4. It recursively retrieves each of the single-cardinality children, regardless whether the child business object is contained with or without ownership.

   Note: All single-cardinality child business objects are processed based on their occurrence in the business object and before the parent business object is processed. Child object ownership and non-ownership do not determine the processing sequence, but they do determine the type of processing.

RetrieveAll operation

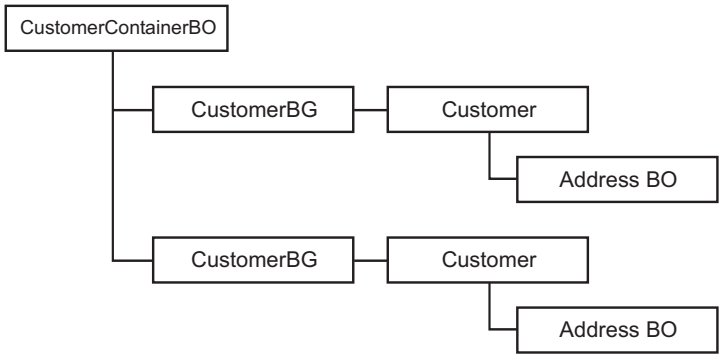
This operation enables the adapter to retrieve an array of business objects from the database.

All of the key and non-key attributes populated in the incoming business object determine the selection criteria. The adapter may retrieve multiple rows for the top-level business object from the database, depending on the attributes selected. If no attributes are populated in the incoming business object, all the rows are retrieved from the respective table in the database.

The adapter performs the following steps to retrieve an array of business objects:

1. For each of the rows retrieved from the database, the adapter constructs a top-level business graph and creates a container of business graphs using all of the retrieved rows. The name of the container business graph is *BOName + ContainerBG*.
2. The adapter retrieves each of the business graphs in the container using the retrieve operation.

The following figure describes the structure of the business object that is returned in a RetrieveAll operation.



The following errors can result from a RetrieveAll operation:

- If any populated business object in the input object does not exist in the EIS, the adapter returns the error `RecordNotFoundException`.
- If the number of hits in the EIS exceeds the value of ResultSetLimit defined in the interaction specification, the adapter returns the error MatchesExceededLimitException. The MatchCount property contains the actual number of hits that the adapter had in the EIS, so that you can either increase the limit, or refine the search appropriately.

  Note: If ResultSetLimit is set to a very large number, problems may occur related to a lack of sufficient memory, depending upon the size and number of business objects returned.
- If any unrecoverable errors are reported by the EIS, the adapter returns the error EISSystemException.

Update operation

The update operation is done by comparing the incoming business object with a business object that is retrieved from the database using the primary keys specified in the top-level, incoming business object.

The adapter performs the following steps when updating a hierarchical business object:

1. It uses the primary key values of the source business object to retrieve the corresponding entity from the database. The retrieved business object is an accurate representation of the current state of the data in the database.

   If the retrieval fails, indicating that the top-level business object does not exist in the database, the adapter returns a `RecordNotFoundException` error, and the update fails.

   If the retrieval succeeds, the adapter compares the retrieved business object to the source business object to determine which child business objects require changes in the database. The adapter does not, however, compare values in the

source business object's simple attributes to those in the retrieved business object. The adapter updates the values of all non-key simple attributes.

If all of the simple attributes in the top-level business object represent keys, the adapter cannot generate an update query for the top-level business object. In this case, the adapter logs a warning and continues to step 2.

2. It recursively updates all single-cardinality children of the top-level business object.

   If the business object definition requires that an attribute represent a child business object, the child must exist in both the source business object and the retrieved business object. If it does not, the update operation fails, and the adapter returns an error.

   The adapter handles single-cardinality children contained with ownership in one of the following ways:

   - If the child is present in both the source and the retrieved business objects, instead of updating the existing child in the database, the adapter deletes the existing child and creates the new child.
   - If the child is present in the source business object but not in the retrieved business object, the adapter recursively creates it in the database.
   - If the child is present in the retrieved business object but not in the source business object, the adapter recursively deletes it from the database.

   For single-cardinality children contained without ownership, the adapter attempts to retrieve every child from the database that is present in the source business object. If it successfully retrieves the child, the adapter populates the child business object but does not update it, because single-cardinality children contained without ownership are never modified by the adapter.

3. It updates all simple attributes of the retrieved business object, except those whose corresponding attribute in the source business object is not specified.

   Because the business object being updated must be unique, the adapter verifies that only one row is processed as a result. It returns an error if more than one row is returned.

4. It processes each multiple-cardinality child of the retrieved business object in one of the following ways:

   - If the child exists in both the source and the retrieved business objects' arrays, the adapter recursively updates it in the database.
   - If the child exists in the source array but not in the retrieved business object's array, the adapter recursively creates it in the database.
   - If the child exists in the retrieved business object's array but not in the source array, the adapter recursively deletes it from the database unless the application-specific information for the attribute that represents the child in the parent has KeepRelationship set to `true`. In this case, the adapter does not delete the child from the database.

UpdateWithDelete operation

This is a special form of the Update operation that can be used to provide better performance than the Update operation.

UpdateWithDelete requires a ChangeSummary. The *ChangeSummary* needs to include business object-level creations and deletions. This enables the adapter to perform operations without the overhead of having to retrieve the existing entities from the EIS and without having to do comparisons, because the ChangeSummary indicates what needs to be done.

If the ChangeSummary is empty, the adapter does not take any action on the request.

DeltaUpdate operation

If the operation in the InteractionSpec is ApplyChanges, and the verb does not exist in the business graph, the adapter performs the DeltaUpdate operation. The adapter inspects the ChangeSummary to identify the operation for each business object in the input hierarchy and performs the operation that was identified.

DeltaUpdate operations are different from Update operations as follows:

- In a DeltaUpdate operation, no retrieve operation occurs before updating.
- No comparisons are made between the incoming business object and the business object in the database.
- All children are processed based on the verb set in each child object. If a child does not have a verb set in it, the adapter returns an error.

The adapter performs the following steps when updating a hierarchical business object with DeltaUpdate. It processes only object changes from the ChangeSummary:

1. It recursively processes all single-cardinality children of the parent object. If a child is marked required in the business object specification, it must be present in the inbound object. If it is not, the DeltaUpdate operation fails, and the adapter returns an error.
2. It sets all foreign key values in the parent that reference attributes in single-cardinality children to their corresponding child values. This is necessary because single-cardinality children might have been added to the database during the previous steps, resulting in the generation of new sequence values.
3. It updates the current object being processed using an SQL Update statement or a stored procedure. All simple attributes of the individual business object are updated. The adapter does not use property level changes to determine which attributes need to be added to the update statement; they are all updated. Because the object being updated should be unique, the adapter checks to ensure that only one row is processed as a result. An error is returned if more than one row is processed.
4. It sets all foreign key values in all cardinality N children of the current object that reference parent attributes to the corresponding parent values. Usually these values are already cross-referenced during data mapping; however, this might not be the case for new children in cardinality N containers. This step ensures that the foreign-key values in all cardinality N children are correct before those children are updated.
5. It updates all cardinality N containers of the current object.

   When the child objects are processed, each child's verb is taken and the appropriate operation is performed. The allowed operations on a child in DeltaUpdate are create, delete, and update:

   - If a Create verb is found in the child, the child is created in the database if it is an ownership child. Non-ownership children are retrieved to validate their existence in the database.
   - If a Delete verb is found in the child, that child is deleted.
   - If an Update verb is found in the child, the child gets updated in the database.

Delete operation

The delete operation is performed by pruning the incoming business object and then retrieving the complete business object from the database. The delete operation is then applied recursively on each business object in the hierarchy.

The delete operation supports physical and logical deletes, depending on the StatusColumnName value in the object's application-specific information. If the StatusColumnName value is defined, the adapter performs a logical delete operation. If the StatusColumnName value is not defined, the adapter performs a physical delete operation.

For physical deletes the adapter takes the following actions:

- It recursively deletes all multiple-cardinality child business objects.
- It deletes the top-level business object.
- It recursively deletes all single-cardinality child business objects contained with ownership.

For logical deletes the adapter takes the following actions:

- It issues an Update that sets the business object's status attribute to the value specified by the business object's application-specific information. The adapter ensures that only one database row is updated as a result, and it returns an error if this is not the case.
- It recursively logically deletes all single-cardinality children contained with ownership and all multiple-cardinality children. The adapter does not delete single-cardinality children contained without ownership.

ApplyChanges operation

The ApplyChanges operation covers a variety of things. It enables any business object requiring a create, update, or delete operation to be processed accordingly by the adapter.

If the top-level verb exists in the business object, then the business object is processed as an after-image. If no top-level verb exists in the business object, then the ChangeSummary is processed.

Application-specific information

Application-specific information in business object definitions provides the adapter with application-dependent instructions on how to process business objects. The adapter parses the application-specific information from the attributes or verb of a business object or from the business object itself to generate queries for create, update, retrieve, and delete operations.

The adapter stores some of the business object's application-specific information in cache and uses this information to build queries for all the verbs.

In an extended or modified application-specific business object, the application-specific information in the business object definition must match the syntax that the adapter expects.

Related tasks

"Querying for database objects" on page 42
After configuring the connection properties, you can run a query for database objects. You can browse the metadata tree structure to understand the structure of objects in the enterprise information system (EIS), and make selections of objects needed for the service description.

Business object naming conventions

Business object names should reflect the structure they represent, such as Customer or Address. Names will most likely be derived during the metadata import process of enterprise metadata discovery, based on the name given by the enterprise information system (EIS).

Business object names should be converted to camel case, in which separators like spaces and underscores are removed, and the first letter of each word is capitalized; for example, ORDER_LINE_ITEM would be converted to OrderLineItem.

The parent business object graph should be named for the contained business object, followed by BG; for example, CustomerBG for a Customer business object.

Business object names have no semantic value to the adapter or the database.

Business objects carry database-specific metadata. They can have a string like JDBC or %AppName% as a prefix to help distinguish between the two types of business objects: application-specific and generic. The remainder of the name can describe the table or stored procedure that the business object represents. For example, if the business object definition is generated for the Employee Table in a database application, such as Human Resources (HR), the respective business object name will be HREmployee.

Application-specific information at the business-object level

Application-specific information at the business object level is used to specify the name of the corresponding database table and to provide information necessary to perform a physical or logical delete operation.

At the business object level, application-specific information format consists of xml that is defined in the jdbcasi.xsd schema definition, where

- *TableName* identifies the database table associated with the business object,
- *StatusColumnName* is the name of the database column used to perform logical delete operations, and
- *StatusValue* is the value that signifies that a business object is inactive or deleted.

For example, assume that a Customer business object has the following value specified for its business object application-specific information:

```
<jdbcasi:TableName>customer</jdbcasi:TableName>
<jdbcasi:StatusColumnName>status</jdbcasi:StatusColumnName>
<jdbcasi::StatusValue>deleted</jdbcasi:StatusValue>
```

Assume that the adapter receives a request to delete a customer. Such a request causes the adapter to issue the following SQL statement:

```
UPDATE customer set status = 'deleted' where pkey = . . . .
```

If the StatusColumnName is not included or no value is specified for it, the adapter physically deletes the business object from the database. In other words, if the business object includes the StatusColumnName parameter in its application-specific information, the adapter performs a logical delete operation. If the business object does not include the StatusColumnName parameter in its application-specific information, the adapter performs a physical delete operation.

Both update and delete operations can use the value of the StatusColumnName property:

- To logically delete the child data, the adapter uses the value of its StatusColumnName parameter to obtain the name of the status column and the text of the status value. For more information, see "Update operations."
- When performing a delete operation, the adapter uses the value of its StatusColumnName parameter to determine whether to physically or logically delete the entire business object. If the StatusColumnName parameter contains a value, the adapter performs a logical delete operation. If the StatusColumnName parameter does not contain a value, the adapter performs a physical delete operation. For more information, see "Delete operations."

Parameters of the ASI that are enabled for use with bidirectional languages are TableName and StatusColumnName. The format for these parameters are transformed based on the attributes set for the BiDi.Metadata property. For more information on this property, see "Configuration properties" on page 68.

Verb application-specific information

The adapter updates database tables using SQL queries or stored procedures, which are groups of SQL statements, as specified in the business objects. Stored procedures and the elements of a stored procedure definition are described in this section. A sample of a stored procedure definition is included.

Related concepts

"Object selection and generation" on page 35
To generate business objects, you select database object nodes. Then the enterprise service discovery wizard generates business objects for the objects of the selected nodes.

Related reference

"Business object attributes and their application-specific information" on page 37
The attributes of a business object are built from the list of columns in the database object. The enterprise service discovery wizard sets the attribute name to the name of the column. Globalized characters are supported in the attribute names. The adapter adds the attribute name, type, and application-specific information.

Stored Procedure overview:

A stored procedure is a group of SQL statements that form a logical unit and perform a particular task. A stored procedure encapsulates a set of operations or queries for the adapter to run on an object in a database server.

The adapter can use simple SQL statements for select, update, retrieve, delete or retrievall operations. The column names for SQL statements are derived from an attribute's AppSpecificInfo property. The WHERE clause is constructed using key values specified in the business object. Each query spans one table only, unless posted to a view.

The adapter calls stored procedures in the following circumstances:

- Before processing a business object, to perform preparatory operational processes
- After processing a business object, to perform post-operational processes
- To perform a set of operations on a business object, instead of using a simple Create, Update, Delete, Retrieve, or RetrieveAll statement.

When it processes a hierarchical business object, the adapter can use a stored procedure to process the top-level business object or any of its child business objects. However, each business object or array of business objects must have its own stored procedure.

Stored procedure definition:

Stored procedures are defined at the verb level. Each stored procedure definition consists of the following elements: StoredProcedureType, StoredProcedureName, ResultSet, and Parameters.

StoredProcedureType defines the type of stored procedure to be used, and this determines when the stored procedure is called. The values are as follows:

- BeforeCreateSP
- AfterCreateSP
- CreateSP
- BeforeUpdateSP
- AfterUpdateSP
- UpdateSP
- BeforeDeleteSP
- AfterDeleteSP
- DeleteSP
- BeforeRetrieveSP
- AfterRetrieveSP
- RetrieveSP
- BeforeRetrieveAllSP
- AfterRetrieveAllSP
- RetrieveAllSP

Note: Stored procedure types associated with RetrieveAll apply to top-level business objects only.

StoredProcedureName is the name of the stored procedure that is associated with the appropriate StoredProcedureType. It is enabled for use with bidirectional languages.

ResultSet determines if the stored procedure returns a result or not (`true|false`). If the result set is returned, an N-cardinality child for the current business object is created using the values returned in the result set rows.

Parameters can be a combination of input only (IP), output only (OP), and input and output (IO). In the case of Oracle stored procedures, a result set can be returned only as an output parameter. In that case, one of the values in the list of parameters is result set (RS). Parameters are enabled for use with bidirectional languages.

Here is a sample of a stored procedure definition:

```
<jdbcasi:JDBCBusinessObjectTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
            <jdbcasi:TableName>customer</jdbcasi:TableName><jdbcasi:Operation>
               <jdbcasi:Name>Retrieve</jdbcasi:Name>
               <jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>RetrieveSP</jdbcasi:StoredProcedureType>
```

```
<jdbcasi:StoredProcedureName>retrieve_cust</jdbcasi:StoredProcedureName>
<jdbcasi:ResultSet>false</jdbcasi:ResultSet>
                     <jdbcasi:Parameters>
                          <jdbcasi:Type>IP</jdbcasi:Type>
                          <jdbcasi:PropertyName>primaryKey</jdbcasi:PropertyName>
                    </jdbcasi:Parameters>
                    <jdbcasi:Parameters>
                          <jdbcasi:Type>OP</jdbcasi:Type>
                           <jdbcasi:PropertyName>custCode</jdbcasi:PropertyName>
                    </jdbcasi:Parameters>
                    <jdbcasi:Parameters>
                          <jdbcasi:Type>OP</jdbcasi:Type>
                           <jdbcasi:PropertyName>firstName</jdbcasi:PropertyName>
                    </jdbcasi:Parameters>
                    <jdbcasi:Parameters>
                          <jdbcasi:Type>OP</jdbcasi:Type>
                              <jdbcasi:PropertyName>lastName</jdbcasi:PropertyName>
                    </jdbcasi:Parameters>
                    </jdbcasi:StoredProcedures>
                    <jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>AfterRetrieveSP</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>retrieve_cust</jdbcasi:StoredProcedureName>
                    <jdbcasi:ResultSet>false</jdbcasi:ResultSet>
                    <jdbcasi:Parameters>
                         <jdbcasi:Type>IP</jdbcasi:Type>
                           <jdbcasi:PropertyName>primaryKey</jdbcasi:PropertyName>
                    </jdbcasi:Parameters>
                    <jdbcasi:Parameters>
                         <jdbcasi:Type>OP</jdbcasi:Type>
                          <jdbcasi:PropertyName>custCode</jdbcasi:PropertyName>
                    </jdbcasi:Parameters>
                    <jdbcasi:Parameters>
                         <jdbcasi:Type>OP</jdbcasi:Type>
                          <jdbcasi:PropertyName>firstName</jdbcasi:PropertyName>
                    </jdbcasi:Parameters>
                    <jdbcasi:Parameters>
                         <jdbcasi:Type>OP</jdbcasi:Type>
                         <jdbcasi:PropertyName>lastName</jdbcasi:PropertyName>
                    </jdbcasi:Parameters>
                  </jdbcasi:StoredProcedures>
              </jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
```

The property ReturnDummyBOForSP returns output parameters even when the result set is `true` but empty. In the case of RetrieveSP, a result set is returned. If the result set is empty, no business objects are created and there is no way to retrieve the output parameters returned by the procedure call. If ReturnDummyBOForSP is `true`, a dummy business object with values from the output and input/output parameters populated in the corresponding attributes is returned. The default value for this property is `false`.

Related concepts

"Business object structure" on page 7
Each business object corresponds to a database table or view, and each simple attribute within the object corresponds to a column in that table or view.

Attribute application-specific information

This topic describes application-specific information (ASI) for attributes and lists the supported parameters with their descriptions.

The application-specific information for attributes differs depending on whether the attribute is a simple attribute or an attribute that represents a child or an array

of child business objects. The application-specific information for an attribute that represents a child also differs depending on whether the parent-child relationship is stored in the child or in the parent.

Application-specific information for simple attributes

For simple attributes, the format for application-specific information consists of a number of parameters and their values. The format of attribute application-specific information is shown in the following example section of an .xsd file:

```
      <jdbcasi:ColumnName>pkey</jdbcasi:ColumnName>
      <jdbcasi:PrimaryKey>true</jdbcasi:PrimaryKey>
      <jdbcasi:FixedChar>true</jdbcasi:FixedChar>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
<simpleType>
   <restriction base="string">
   <maxLength value="10"/>
   </restriction>
</simpleType>
</element>
<element name="custCode" type="string">
<annotation>
<appinfo source="WBI">
<jdbcasi:JDBCAttributeTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:ColumnName>ccode</jdbcasi:ColumnName>
   <jdbcasi:ForeignKey>custinfoObj/custCode</jdbcasi:ForeignKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="firstName" type="string">
<annotation>
<appinfo source="WBI">
<jdbcasi:JDBCAttributeTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:ColumnName>fname</jdbcasi:ColumnName>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="lastName" type="string">
<annotation>
<appinfo source="WBI">
<jdbcasi:JDBCAttributeTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:ColumnName>lname</jdbcasi:ColumnName>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
```

The only required parameter for a simple attribute to be processed by the connector is the column name. For example, this is the format to specify only the column name, where `ccode` stands for customer code:

```
<jdbcasi:ColumnName>ccode</jdbcasi:ColumnName>
```

The attribute ASI parameters that are enabled for use with bidirectional languages are ColumnName and ForeignKey. The format of these parameters is transformed based on the attributes set for the BiDi.Metadata property. For more information on this property, see ″Configuration properties″ in the ″Reference″ section. For more information on bidirectional properties, see the bidirectional support general technical paper and the adapter technical paper on the IBM developerWorks(R) Web site.

The table titled, “Parameters in attribute application-specific information” lists each parameter and its description.

Parameters in attribute application-specific information

Parameter	Description
ByteArray	If `true`, the adapter reads and writes binary data to the database and sends that data as a string to the application server. By default the value is `false`. For more information, see "Working with binary data."
ColumnName	The value of this parameter is the name of the database column for this attribute. This is enabled for use with bidirectional languages.
FixedChar	This parameter specifies whether the attribute is of fixed length when the columns in the table are of type CHAR, not VARCHAR. For example, if a particular attribute is linked to a column that is of type CHAR, the adapter pads the attribute value with blanks to the maximum length of the attribute when querying the database. By default the value is `false`. This parameter needs to be updated manually in the business object .xsd file. You can edit the file either in text mode or by using the Business Object Editor in WebSphere Integation Developer. Ensure that no validation errors occur in the .xsd after it has been updated. See the code example for this parameter following this table.
ForeignKey	The value of this property depends on whether the parent/child relationship is stored in the parent business object or in the child. If it is stored in the parent, set the value to include both the type of the child business object and the name of the attribute in the child to be used as the foreign key *(ChildBOname/ChildPropertyName)*. If it is stored in the child, set the value to include only the name of the attribute in the parent to be used as the foreign key. If an attribute is not a foreign key, do not include this parameter in the application-specific information. This parameter is enabled for use with bidirectional languages.
KeepRelationship	If `true`, this parameter prevents the deletion of a child business object during an update operation.

Parameter	Description
OrderBy	If a value is specified for this parameter and the attribute is in a child business object, the adapter uses the value of the attribute in the ORDER BY clause of retrieval queries. The adapter can retrieve child business objects in either ascending order (ASC) or descending order (DESC). If you do not include this parameter in the application-specific information, the adapter does not use this attribute when specifying retrieval order.
Ownership	This parameter specifies that a child business object is owned by the parent. If `true`, then create, update, and delete operations on the child business object are allowed. If `false`, then none of these updates can be applied to the child business object. When its parent is created, the existence of the child is validated to ensure that relationship integrity is maintained in the database.
PrimaryKey	If the value is `true`, this implies that the column associated with this attribute is a key in the corresponding table in the database.
UniqueIdentifier (UID)	The adapter uses this parameter to generate the unique ID for the business object. It supports the generation of sequences and identity columns *(UID=AUTO|SequenceName)*. Sequences can be defined for DB2 and Oracle databases only. Identity columns can be defined for DB2 and Microsoft SQL Server. If the attribute does not require a unique ID, do not include this parameter in the application-specific information.

Example of FixedChar parameter in the business object .xsd file

```
<element name="primaryKey">
<annotation>
<appinfo source="WBI">
        <jdbcasi:JDBCAttributeTypeMetadata
xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
                <jdbcasi:ColumnName>pkey</jdbcasi:ColumnName>
                <jdbcasi:PrimaryKey>true</jdbcasi:PrimaryKey>
                <jdbcasi:FixedChar>true</jdbcasi:FixedChar>
        </jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
<simpleType>
        <restriction base="string">
                <maxLength value="10"/>
        </restriction>
</simpleType>
</element>
```

Related reference

"Business object attributes and their application-specific information" on page 37

The attributes of a business object are built from the list of columns in the

database object. The enterprise service discovery wizard sets the attribute name to the name of the column. Globalized characters are supported in the attribute names. The adapter adds the attribute name, type, and application-specific information.

Installing the adapter

Information in this section covers the installation requirements for the Adapter for JDBC and describes the installed file structure.

For information on how to install the adapter, see Installing IBM WebSphere Adapters.

Adapter environment

Hardware and software requirements of the adapter environment are available online.

To find the requirements for this adapter, refer to "IBM WebSphere Adapters and IBM WebSphere Business Integration Adapters: software requirements" at http://www-1.ibm.com/support/docview.wss?uid=swg27006249. Select your adapter from the list of WebSphere adapters.

Installation information specific to the Adapter for JDBC

The Adapter for JDBC has installation prerequisites regarding the JDBC driver and the setup of the event store.

JDBC driver prerequisites

You need to set up the JDBC driver so that the JDBC driver JAR file is in the class path.

JDBC enterprise service discovery exists within a project in the enterprise service discovery wizard. The JDBC driver JAR file needs to be in the class path of the JDBC project in this tool to enable you to run the JDBC enterprise service discovery process.

Event store set up

You need to set up the event store in the database before inbound processing can be done. Sample scripts are provided to set up the event store for the IBM® DB2®, Oracle, or Microsoft™ SQLServer database, as follows:

- `WBIA_JDBC_EventStore_DB2.sql`
- `WBIA_JDBC_EventStore_Oracle.sql`
- `WBIA_JDBC_EventStore_MSSQL.sql`

You should set up triggers on user tables as needed so that changes to the user tables can automatically generate events that are stored in the event store.

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 CWYBC_JDBC.rar file has the following absolute path: `c:\WebSphereBI.\adapter\JDBC\deploy\CWYBC_JDBC.rar`

The resource adapter archive (RAR) file contains both the adapter and the enterprise service discovery tool files.

UNIX(R) and Windows(TM) 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(TM) directories and files for the WebSphere Adapter for JDBC. Directories and files are grouped into categories.

File and directory category	Directories and files
Adapter for JDBC RAR file	adapter/deploy/CWYBC_JDBC.rar
IBM Support Assistant plug-in zip file	adapter/ISAPlugin/com.ibm.esupport.client.SS6FE6_RAJDBC.zip
IBM Tivoli License Manager (ITLM) file	adapter/5724L77E060000.sys
Message files	adapter/messages/CWYBC_JDBC_messages.tar
Message files for Foundation Classses	adapter/messages/CWYBS_AdapterFoundation_messages.tar
Notices file for ICU4J	adapter/notices.txt
Sample application EAR file	adapter/samples/Apps/JDBCApp.ear
Sample scripts for IBM DB2	adapter/samples/scripts/scripts_db2.sql
Sample scripts for Oracle	adapter/samples/scripts_oracle.sql

Directory and file structure for Windows

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

File and directory category	Directories and files
Adapter for JDBC RAR file	adapter\deploy\CWYBC_JDBC.rar
IBM Support Assistant plug-in zip file	adapter\ISAPlugin\com.ibm.esupport.client.SS6FE6_RAJDBC.zip
IBM Tivoli License Manager (ITLM) file	adapter\5724L77E060000.sys
Message files	adapter\messages\CWYBC_JDBC_messages.zip
Message files for Foundation Classses	adapter\messages\CWYBS_AdapterFoundation_messages.zip
Notices file for ICU4J	adapter\notices.txt
Sample application EAR file	adapter\samples\Apps\JDBCApp.ear
Sample scripts for IBM DB2	adapter\samples\scripts\scripts_db2.sql
Sample scripts for Oracle	adapter\samples\scripts_oracle.sql

Creating the adapter project

You need to create an adapter project in WebSphere(R) Integration Developer. You then generate business objects and service constructs using the enterprise service discovery wizard in WebSphere Integration Developer. You set configuration properties. Finally, you deploy the adapter project to the application server.

Before you deploy the adapter, you must install these products:

- WebSphere Integration Developer, Version 6.0
- WebSphere Process Server--You will use the WebSphere Process Server administrative console to deploy the adapter project on the application server and to reconfigure property values.

Refer to the IBM(R) WebSphere Process Server installation instructions at http://www.ibm.com/software/integration/wps.

WebSphere Adapters can only be installed on computers with Windows(TM) or Linux(TM) operating systems. From there they can be deployed to UNIX(R)-based systems. Each adapter is installed as a resource adapter archive (RAR) file.

In principle, deploying an adapter is the same as deploying any other component on the WebSphere Process Server. For more information on deploying components on WebSphere Process Server, refer to the WebSphere Integration Developer user guide at http://www.ibm.com/software/integration/wid.

Creating the adapter project requires that you perform the following tasks:

- Create a project for the adapter
- Add vendor libraries
- Generate business objects and service constructs
- Set configuration property values, and save the values and artifacts in a new business integration module
- Deploy the adapter project to the server, and start the application

Creating a project for the adapter

The first task in deploying the adapter is to create a J2EE(TM) connector project for the adapter. The resource adapter archive (RAR) file for the Adapter for JDBC needs to be imported into WebSphere(R) Integration Developer. This sets up the project in your workspace in WebSphere Integration Developer.

1. Start WebSphere Integration Developer

   For details, refer to the WebSphere Integration Developer user guide at http://www/ibm.com/software/integration.wid.

2. Import the RAR file

   In WebSphere Integration Developer go to the Business Integration perspective. Click **File > Import**. In the Select window, select **RAR file** as the import source and click **Next**.

   In the Connector Import window, use the **Browse** button to select the location of the RAR file. A project name automatically appears in the **Connector project** field, but you can change it if you want.

   Deselect the check box **Add module to an EAR project**.

   Click **Finish** to import the RAR file. This creates a J2EE Connector project in the workspace.

3. Add the JDBC driver to the connector project

You need to add the JDBC driver to the connector project for it to become part of the enterprise application archive (EAR) file that you will deploy to the application server. You can do this either after you import the RAR file, or after you install the EAR on the application server.

After you import the RAR file, you would add the JAR file to the appropriate folder in the workspace. For example, the JAR file could be added to this location: `C:\workspace\CWYBC_JDBC\connectorModule`

Instead, if you want to add the JDBC driver to the connector project after you install the EAR on the application server, then after installing the application you would add the JAR file to the RAR subdirectory of the installedApps directory of WebSphere Process Server. See "Deploying the adapter project" for details on installing the application on the application server.

Related tasks

"Deploying the adapter project" on page 48
Your project file is a J2EE(TM) Connector project in your workspace in WebSphere(R) Integration Developer. You need to export it to your local file system as an enterprise application archive (EAR) file. Then you need to provide authentication information for JCA connector security before you upload and install the EAR file for the project on the application server.

Adding vendor libraries

After you create the project in WebSphere(R) Integration Developer, you need to add a reference to the JDBC driver to your project.

Add the JDBC driver to the Java build path

In WebSphere Integration Developer at the Configuration window, right click **Connector Project**. Select **Properties**.

To add the external JAR file, click **Java Build Path**. Select the **Libraries** tab and click **Add External Jars**. In the File System window, navigate to the **JDBC Driver** and select the JAR file.

Generating business objects

To generate business objects, you first set connection properties. Then you run a query for database objects so that you can select objects needed for the service description. Next you need to specify values for the Selection properties for the import and export files. Finally, you set configuration properties, and save the artifacts and property values in a new business integration module.

Before beginning the tasks to generate business objects, you can read details about the process in the sections about metadata import, discovery of system capabilities, and data descriptions.

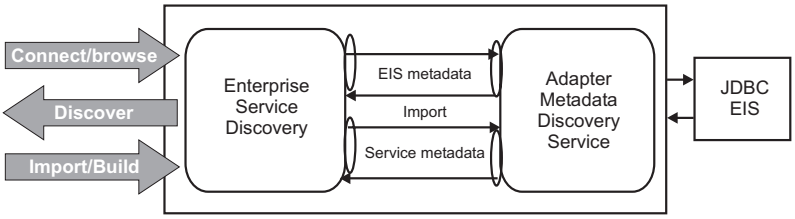
Metadata import

You use the enterprise service discovery wizard in the WebSphere(R) Adapter for JDBC to discover objects in a database and to generate business objects from the selected objects. Enterprise service discovery support also generates the service constructs that enable the adapter to run as a Service Component Architecture (SCA) component.

Objects from which business objects can be created include tables, views, stored procedures, and synonyms/nicknames. The adapter enables you to discover the objects by generating a list of all the schemas in the database. Within each schema

are lists of tables, views, stored procedures, and synonyms/nicknames that you can select and request the enterprise service discovery wizard to generate the corresponding business objects. The wizard analyzes the metadata of objects and generates attributes in the business object. The attributes are generated based on the column names for all database objects. To summarize its functions, the enterprise service discovery wizard:

- Generates a business object corresponding to a database object.
- Generates properties in the business object corresponding to the properties in the database object.
- Sets application-specific information on the business object.
- Provides service descriptions (inbound and outbound) that are used to generate the import, export and Web Services Description Language (WSDL) files.



ESD high-level diagram

Discovery of system capabilities

The adapter discovers business objects in a database by analyzing the database to identify the schemas. Then it creates a list of all the objects from the database and shows it as a tree structure.

The schemas are displayed as the top-level nodes in the tree. The schema nodes are not selectable for generation.

Under each schema are nodes labeled Tables, Views, Stored Procedures, and Synonyms/Nicknames. These nodes are not selectable for generation. The objects listed under these nodes are the names of the tables, views, stored procedures, and synonyms/nicknames. These nodes are marked as selectable for generation. If no tables, views, stored procedures, or synonyms exist for a particular schema, none are listed.

You can specify filter properties if you want to narrow the list of schemas displayed in the tree; otherwise, all schemas are displayed. “Filter and node properties” in the “Reference” section describes the properties you need to provide.

You can select a tables, views, stored procedures, or synonyms/nicknames node, then click **Filter**, and the enterprise service discovery wizard queries for an ObjectNameFilter. You can use the ObjectNameFilter property to filter the list of database objects to display. “Filter and node properties” in the “Reference” section describes the ObjectNameFilter property.

Related reference

“Filter and Node properties” on page 79
During business object discovery, you can specify Filter properties if you want to narrow the list of schemas displayed in the tree; otherwise, all schemas are displayed. You can set the Node property if you want to narrow the list of database objects to display.

Object selection and generation:

To generate business objects, you select database object nodes. Then the enterprise service discovery wizard generates business objects for the objects of the selected nodes.

You can select multiple database object nodes. When you specify filter properties, if you check **Add business object ASI**, then for each node that you select, the enterprise service discovery wizard requests the object-level parameters for StatusColumnName, StatusValue, and stored procedure association.

For StatusColumnName, you are presented with a list of the actual column names for the particular object from which to set the StatusColumnName. You need to type the StatusValue. These values are set in the business object level application-specific information (ASI).

You can choose to associate stored procedures to the business objects. A list of all the supported stored procedure types is presented. Each stored procedure type has a list of stored procedures available in the database. One stored procedure can be assigned to a particular stored procedure type.

For each stored procedure assigned, a list of the stored procedure's input and output parameters is presented. Each parameter has a list of the business object's attributes. The parameter's type is listed in the property's description. You may select one business object attribute for each stored procedure parameter. All of the stored procedure types that have a stored procedure assigned are added to the verb application-specific information of the business object. See "Verb application-specific information."

After you have selected database objects, you need to set values for the Selection properties. The enterprise service discovery wizard queries for the Selection properties. For details about these properties, see "Selection properties" in the "Reference" section.

Related tasks

"Selection properties" on page 80
After you have selected database objects, you need to set values for the Selection properties.

Related reference

"Verb application-specific information" on page 24
The adapter updates database tables using SQL queries or stored procedures, which are groups of SQL statements, as specified in the business objects. Stored procedures and the elements of a stored procedure definition are described in this section. A sample of a stored procedure definition is included.

Data descriptions

Data descriptions are some of the service constructs that the enterprise service discovery process generates to enable the adapter to run as a Service Component Architecture (SCA) component.

The data description includes a definition of the structure and content of adapter business objects that is passed between the client application and the adapter at run time. The data description enables the client application to create the proper data objects for requests, and to interpret the data objects returned as responses. The data description generated from the database components is represented as an XML schema.

- The business object maps to a complex type definition.
- The attributes of the business object map to element type definitions.
- The application-specific information for the business object is contained in annotations at the complex type.
- The application-specific information for each property in the business object is contained in annotations for the element types.

Note: The template for the application-specific properties for the business object and the attribute level are defined in the metadata schema for the JDBC adapter. The name of the schema file is JDBCASI.xsd. The generated schema file has a reference to this template in its annotations.

Business object schema and its application-specific information:

The business object schema is built out of database components that you select. Each component translates into a top-level business object.

The enterprise service discovery wizard generates the name of the business object in the form of *PrefixSchemaNameObjectName*, where

- *Prefix* is the value as specified in the connection property named Prefix. Prefix is not required, and if not specified, no prefix will be added to the business object name.
- *SchemaName* is the name of the schema to which the object belongs.
- *ObjectName* is the name of the table, view, stored procedure, or synonym/nickname.

Globalized characters are supported in the business object name.

The enterprise service discovery wizard sets the TableName application-specific information attribute to a value in the form of *schemaname.tablename*. It sets the business object level application-specific information as listed in the table "Business object application-specific information (ASI)." The operations you select will be set in the business object. All of the generated business objects have the same structure, regardless of the object type they are generated from, either table, view, stored procedure, or synonym/nickname. All have attributes based on the columns, and the TableName ASI is set to the name of the object.

Business object application-specific information (ASI)

Business object ASI	Set by enterprise service discovery wizard	Additional information
TableName	Yes	Set to the actual name of the column
StatusColumnName	Yes	You specify during object selection
StatusValue	Yes	You specify during object selection

All business objects created are top-level. The enterprise service discovery wizard will not create any recursive (child) business objects. The enterprise service discovery wizard also generates business graphs for all business objects, because all are top-level. The name of the business graph will be the business object name followed by "BG." For example, a business object with the name

JDBCSchema1Customer, would have a the business graph named JDBCSchema1CustomerBG. The operations set in the business object are also set in the business graph.

Business object attributes and their application-specific information:

The attributes of a business object are built from the list of columns in the database object. The enterprise service discovery wizard sets the attribute name to the name of the column. Globalized characters are supported in the attribute names. The adapter adds the attribute name, type, and application-specific information.

The types returned by the JDBC metadata are mapped to the business object attribute types as listed in the table "JDBC metadata column and business object attribute types." Only the JDBC types listed are supported by the adapter. Any columns with types not listed are not added to the business object. An informational message is produced stating, for example, `The column named xxxx in the table named yyyy is not of a supported type and will not be added to the business object.`

JDBC metadata column and business object attribute types

JDBC metadata column type	Business object attribute type
BIT	BOOLEAN
CHAR LONGVARCHAR VARCHAR	STRING
INTEGER NUMERIC SMALLINT TINYINT BIGINT	INTEGER
TIME TIMESTAMP DATE	DATE
DECIMAL	STRING
DOUBLE FLOAT	DOUBLE
REAL	FLOAT

The table titled "Attribute information" lists the attribute information set by the enterprise service discovery wizard and describes how it is set.

Attribute information

Attribute information	Set by enterprise service discovery	Additional information
Cardinality	No	
Name	Yes	Name of the attribute. This is enabled for bidirectional languages.

Attribute information	Set by enterprise service discovery	Additional information
MinOccurs/MaxOccurs	Yes	If the column is not a primary key and is not nullable, the attribute is required, and the values for the attribute will be set to at least 1.
Type	Yes	Set as shown in the table titled "JDBC metadata column and business object attribute types."

The enterprise service discovery wizard sets the attribute application-specific information (ASI) in the business object as shown in the table titled "Attribute application-specific information." For more information on the attribute application-specific information see "Application-specific information for simple attributes."

Attribute application-specific information

Attribute ASI	Set by enterprise service discovery	Additional information
ColumnName	Yes	Set to the actual name of the column. This is enabled for bidirectional languages.
FixedChar	No	Needs to be updated manually in the business object .xsd file. Use either text mode or the business object editor in WebSphere Integration Developer to edit the file. After updating the file, ensure there are no validation errors. See an example of FixedChar in an .xsd file in the section "Application-specific information for simple attributes."
ForeignKey	No	
OrderBy	No	
PrimaryKey	Yes	If the column is a primary key, PrimaryKey will be set to `true`.
UID	No	

If you choose to add stored procedures to the business objects, the verb application-specific information (ASI) is set as specified in the table titled "Verb application-specific information." For information on valid stored procedure types, see the section "Verb application-specific information."

Verb application-specific information

Verb ASI or Stored procedure parameters element	Set by enterprise service discovery wizard	Additional information
Parameters	Yes	Lists the stored procedure parameters. This is enabled for bidirectional languages.
PropertyName	Yes	Set to the name of the business object attribute that you select. This is enabled for bidirectional languages.
ResultSet	No	If the stored procedure returns a ResultSet, you need to set this parameter to `true` in the business object definition.
StoredProcedure	Yes	Set to the stored procedure name. This is enabled for bidirectional languages.
StoredProcedure Type	Yes	You choose from a list of types.
Type	Yes	Set to the type of the stored procedure parameter (IP/OP/IO) .

To build hierarchical business objects

The enterprise service discovery wizard generates flat business objects. It does not use the foreign key constraints that are defined in the database between different tables to build relationships automatically. These need to be linked manually. You can update the business object definitions either in text mode or by using the business object editor.

An example of the .xsd definition file for single- and multiple-cardinality child business objects is provided here. The element custInfoObj is a single-cardinality child business object, and addressObj is a multiple-cardinality child business object.

```
<element name="addressObj" minOccurs="1" type="Address:Address"
maxOccurs="unbounded">
                        <annotation>
                        <appinfo source="WBI">
                        <pasi:JDBCAttributeTypeMetadata xmlns:pasi=
"urn:app:jdbc:asi">
                                <pasi:Ownership>true</pasi:Ownership>
                        </pasi:JDBCAttributeTypeMetadata>
                        </appinfo>
                        </annotation>
                        </element>
                        <element name="custInfoObj" minOccurs="0" type=
"CustInfo:CustInfo" maxOccurs="1">
                        <annotation>
                        <appinfo source="WBI">
                        <pasi:JDBCAttributeTypeMetadata xmlns:pasi=
"urn:app:jdbc:asi">
                                <pasi:Ownership>false</pasi:Ownership>
                        </pasi:JDBCAttributeTypeMetadata>
                        </appinfo>
                        </annotation>
                        </element>
```

Related reference

"Attribute application-specific information" on page 26
This topic describes application-specific information (ASI) for attributes and lists the supported parameters with their descriptions.

"Verb application-specific information" on page 24
The adapter updates database tables using SQL queries or stored procedures, which are groups of SQL statements, as specified in the business objects. Stored procedures and the elements of a stored procedure definition are described in this section. A sample of a stored procedure definition is included.

Setting connection properties

After you have created the adapter project, you need to initialize the enterprise service discovery wizard for the Adapter for JDBC and set the values of the connection properties for your database instance.

1. Initialize enterprise service discovery

   In WebSphere Integration Developer, switch to the **Business Integration** perspective. At the **Business Integration** tab that contains the JDBC connector project, which is highlighted, right click in that pane. In the pop-up menu select **New > Enterprise Service Discovery**.

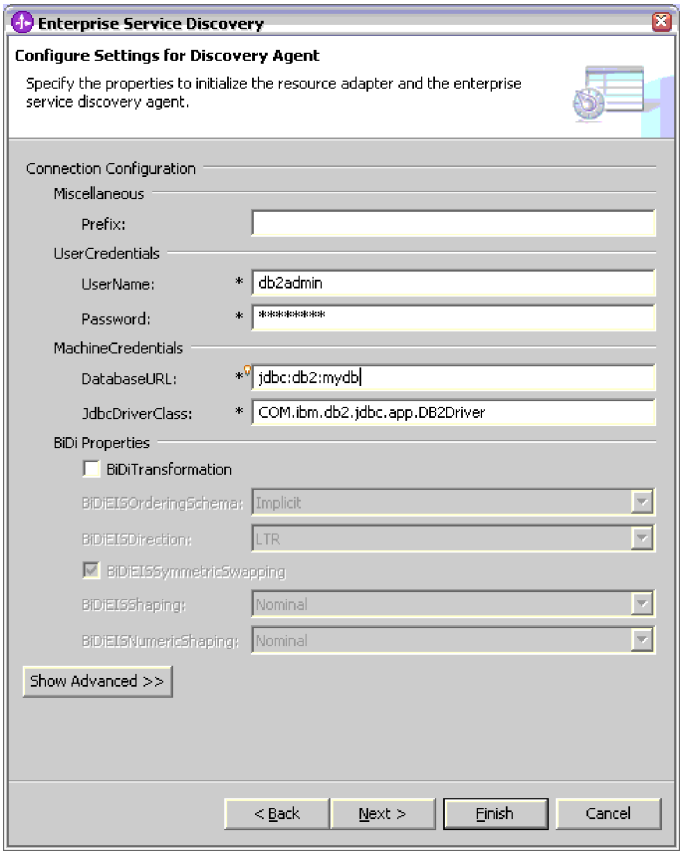
   In the Select an Enterprise Service Resource Adapter window, select the option for your adapter and click **Next**. If you had not already imported the RAR file, you could click **Import Resource Adapter** in this window to import it.

2. Set values of connection properties

   You need to set the values of the Metadata discovery connection properties used to connect to the target EIS instance for discovery and for creating the service description. If you want to enable bidirectional script data processing, you need to activate bidirectional transformation and set values for the bidirectional properties.

   In the Configure Settings for Discovery Agent window, which follows, type the values for the connection configuration properties. For details on these properties, see "Metadata discovery connection properties" in the "Reference" section.

   To activate bidirectional capability, select the check box next to **BiDi Transformation**. Then set the values for the bidirectional properties. For details on these properties, see "Bidirectional connection properties" in the "Reference" section.



Configure settings for Discovery Agent window

3. Select logging options

   If you click **Show Advanced**, the **Logging options** appear. **Logging options** are used to set up logging and tracing for the enterprise service discovery process only. However, the logging and tracing levels are the same as for the adapter. For details on logging and tracing levels for the adapter, see "Enabling Logging" and "Enabling Tracing."

   Under **Logging options**, enter or browse for the output location of the log file. Select the logging and tracing levels for enterprise service discovery. Click **Next**.

The discovery service uses the connection properties to prepare a metadata tree that is displayed for object selection and navigation.

Related tasks

"Enabling logging" on page 52
The WebSphere(R) Adapter for JDBC 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. Because 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.

"Enabling tracing" on page 54
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.

Related reference

"Metadata discovery connection properties" on page 77
The enterprise service discovery process requires these properties to connect to the enterprise information system (EIS) for discovery and for creating the service description.

"Bidirectional connection properties" on page 78
These properties enable the enterprise service discovery wizard to apply the proper bidirectional transformation on the data passed to the enterprise information system (EIS).

Querying for database objects

After configuring the connection properties, you can run a query for database objects. You can browse the metadata tree structure to understand the structure of objects in the enterprise information system (EIS), and make selections of objects needed for the service description.

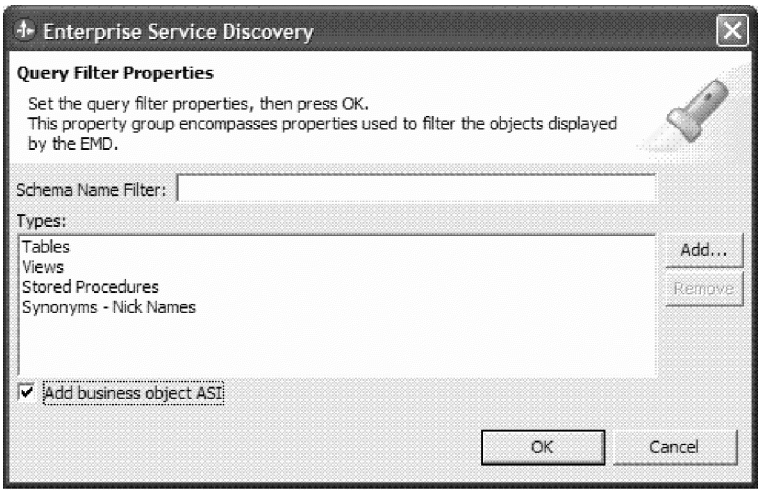
Before running the query, you can specify Filter Properties if you want to narrow the list of schemas, nodes, or objects displayed in the tree structure.

1. Specify Filter Properties

   In the window Find and Discover Enterprise Services, click **Edit Query**. In the pop-up window Query Filter Properties, enter text in the **Schema Name Filter** property field. The schemas that start with the specified string are displayed. Select the schemas that you want to use.

   The **Types** property field lists the entries: tables, views, stored procedures, and synonyms/nicknames. You can add or remove nodes from that list.

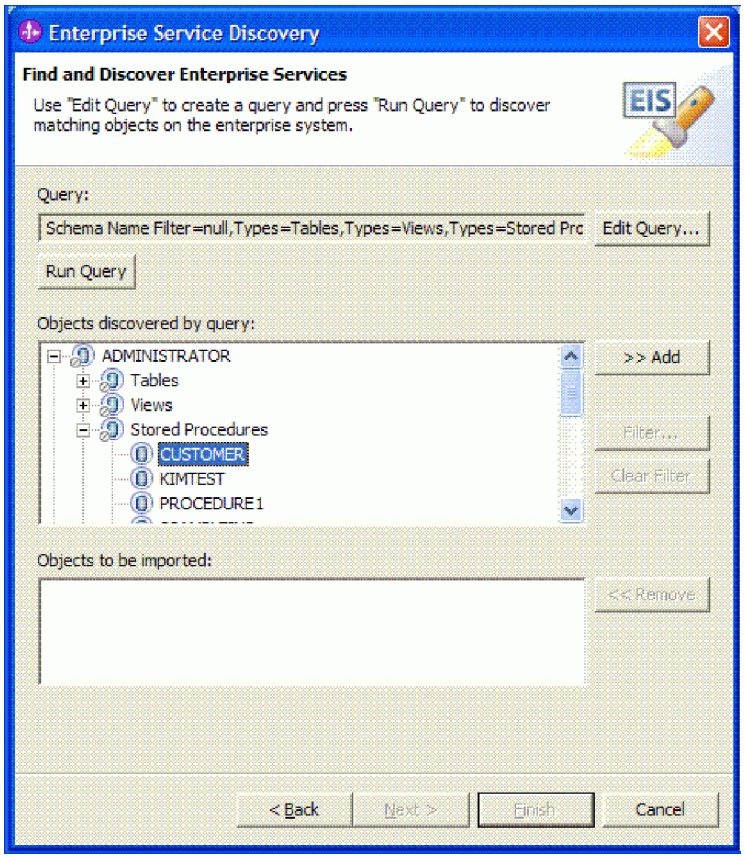
   In the Query Filter Properties window, you can check **Add Business Object ASI**. Then whenever you add an object when you run the metadata query in Step 2, a window called Configuration Parameters for *(name of object)* appears for entering application-specific information.



Query Filter Properties window

2. Run the metadata query
   a. Display objects discovered by query

      In the Find and Discover Enterprise Services window, which follows, click **Run Query**. The objects are displayed in the top pane.



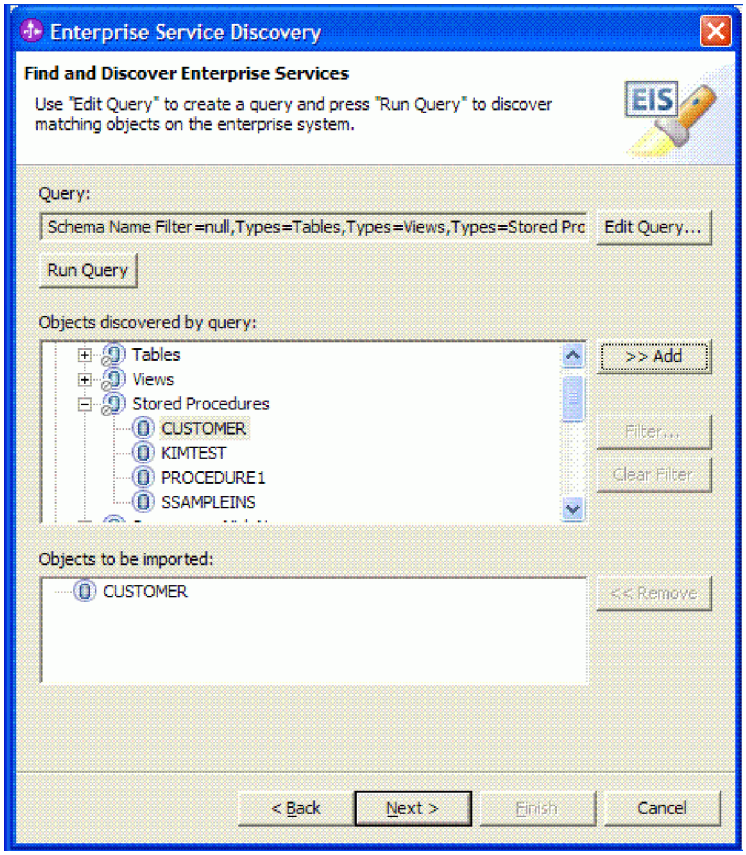
Find and Discover Enterprise Services window

b. Filter objects

You can filter objects similar to the way you filtered schemas. Select a tables, views, stored procedures, or synonyms/nicknames node. Click **Filter**. The enterprise service discovery wizard queries for an **Object Name Filter** to filter the list of database objects to display for that node. Type in text, and only those database objects that start with the specified string are displayed.

c. Select objects for import

Highlight an object and click **Add** to select objects to be imported. The selected objects appear in the bottom pane. To remove a selected object, highlight it and click **Remove**.



Customer object has been selected for import

d. Add business object ASI

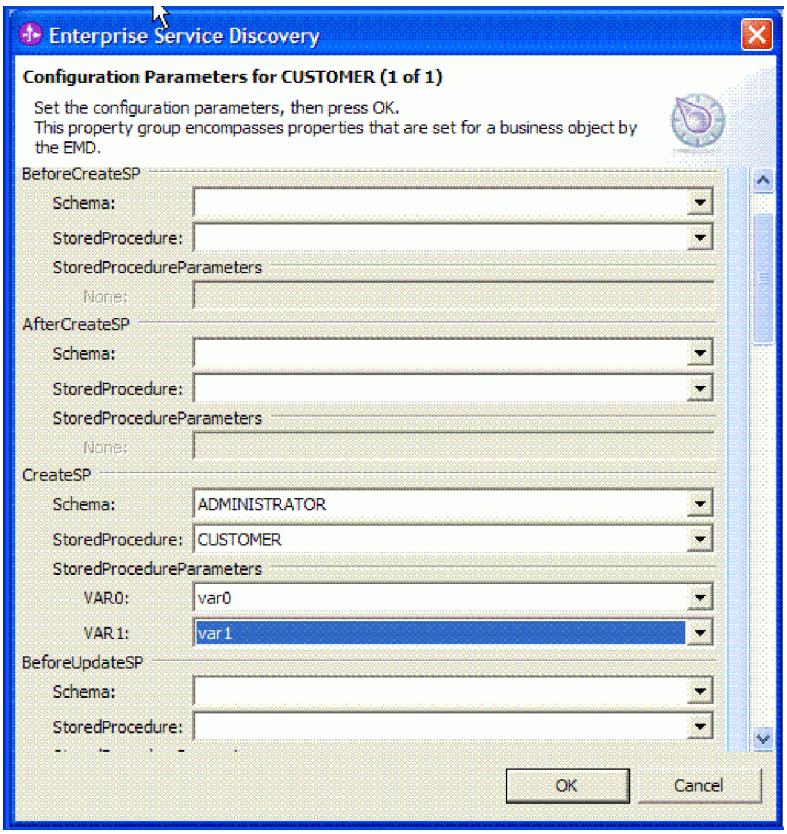
If you checked **Add Business Object ASI** in the Query Filter Properties window, then whenever you add an object, a window called Configuration Parameters for *(name of object)* appears for entering application-specific information. This window is shown here with verb application-specific information parameters for the **CUSTOMER** object.

Select the verb that your stored procedure should be associated with; for example, if your stored procedure creates a record in a table, enter the details for the **CreateSP** configuration parameter. Enter the values for the **Schema** name, **StoredProcedure** name, and **StoredProcedureParameters**.

Click **OK** when you are finished entering ASI.

If you select more than one object to add, the window for the first object appears. After you enter the ASI and click **OK**, the window for the next object appears, and so on. For information on the object-level, verb, and attribute application-specific information, see "Application-specific information."

Click **Next**.



Configuration Parameters for CUSTOMER window

Related tasks

"Deploying and configuring for Scenario 2" on page 63
In Scenario 2, you will set adapter connection properties and generate business objects. You will export your project to an enterprise application archive (EAR) file, deploy the project on the application server, and reset configuration properties.

Related reference

"Application-specific information" on page 22
Application-specific information in business object definitions provides the adapter with application-dependent instructions on how to process business objects. The adapter parses the application-specific information from the attributes or verb of a business object or from the business object itself to generate queries for create, update, retrieve, and delete operations.

Setting Selection Properties

After you have selected database objects, you need to specify values for the Selection properties for the import and export files.

For details on the Selection properties, see the "Reference" section.

1. Specify NameSpace

   NameSpace is initially set to the default NameSpace for all business objects. This default value appears in the Configure objects window. It is the NameSpace for the metadata schema, JDBCASI.xsd.

   NameSpace is prepended to the business object name to keep business object schemas logically separated.

2. Select Service Type

Select either **Inbound** or **Outbound** for the Service Type.

3. Select Operations

   In the Configure objects window, the **Operations** field lists the operations that the adapter supports for the Service Type you selected. If you want to add to the list of these operations, click the **Add** button. For example, you might remove an operation and then decide you want to include it. At the Add window, select from the list of operations and click **OK**. When you are finished, click **Next**.

   The specified operations are set for all business objects being generated.

4. Set MaxRecords

   Enter the maximum number of records to retrieve for a RetrieveAll operation.

5. Specify BOLocation

   Enter the path to the location where the generated .xsd files are to be stored.

   Related tasks

   "Selection properties" on page 80
   After you have selected database objects, you need to set values for the Selection properties.

   "Deploying and configuring for Scenario 2" on page 63
   In Scenario 2, you will set adapter connection properties and generate business objects. You will export your project to an enterprise application archive (EAR) file, deploy the project on the application server, and reset configuration properties.

Saving the adapter project

After you specify selection properties, you configure properties that the adapter uses to set up a communication channel to a specific database. These include Resource adapter, J2C connection factory, J2C activation specification, and bidirectional properties. You also need to create a new business integration module where all the artifacts and property values can be saved.

1. Specify new module name

   In the Generate artifacts window, which is shown below, unless a module name appears in the **Module** field, you need to click **New**. In the New Module pop-up window, enter the module name and click **Finish**.

2. Specify folder for Service Description

   In the Generate Artifacts window, either specify or browse for a folder within the new module where the Service Description will be saved.

   Depending on the Service Type you selected, either the import or export file name appears in the **Name** field. You could add a comment in the **Description** field.

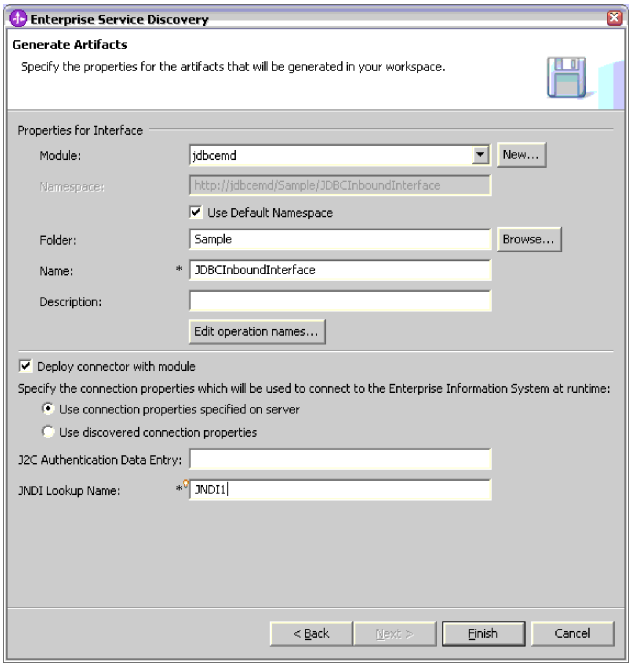
   If you click **Edit Operations**, a pop-up window appears with the default names of all the operations for the business objects you are creating. The default names comprise the operation name combined with the business object name.

   Note: The **Deploy connector with module** check box must be checked to ensure that the connector project RAR file is included in the EAR file that you deploy to the application server.

3. Set configuration property values

   The option **Use connection properties specified on server** can be used in the future after you deploy the project, whenever you need to create new business objects. It indicates that you want to use the properties already on the application server. The **J2C Authentication Data Entry** field is used to specify

an authentication alias when one has already been set up on the application server. The **JNDI Lookup Name** is used to name the connection factory that will be used on the application server.



Generate Artifacts window

To set property values, select **Use discovered connection properties**.

For inbound processing, the property fields appear for the J2C activation specification and Resource adapter properties. For outbound processing, the property fields appear for the J2C connection factory and Resource adapter properties. If you have activated bidirectional transformation, the bidirectional property fields appear for inbound or outbound processing. Required property fields are marked with an asterisk. For details on these configuration properties, see the "Reference" section.

Important: If you specify J2C activation specification properties at this time using the enterprise service discovery wizard, the property settings cannot be changed later. After you deploy the project as a new application on the application server, you will not be able to update the J2C activation specification properties using the WebSphere Process Server administrative console, even though you will be able to change other configuration properties using the administrative console. If you try to update the properties later, the adapter will not recognize the updated values. If you want to set the J2C activation specification properties after installing the project on the application server, do not set those property values now.

Related reference

"Configuration properties" on page 68
This group of properties contains attributes used by the adapter to set up a communication channel to a specific database application.

Generating reference bindings

Reference bindings are used by other WebSphere(R) Business Integration Service Component Architecture (SCA) components to access the adapter. You create a reference to the adapter from the project module to link the adapter to other server processes.

Reference binding generation is required in a testing environment only. It is not necessary when deploying the adapter in a production environment.

1. Create SCA component

   Go to the **Business Integration** perspective of WebSphere Integration Developer. In the **Business Integration** tab, right-click the **JDBCEMD** module, and select **Open With > Assembly Editor**.

   The Assembly Diagram window appears with the module's Import component in view. To create a new component, click the second icon (Import) in the left-hand (vertical) frame of the window. A new menu of icons appears.

   Click the bottom icon in the new menu of icons (it has hover-help that reads **Standalone References**). The cursor changes to the placement icon.

   Click the palette to add the new component to the Assembly Diagram window.

2. Create a standalone reference

   Click and drag the module's Import component to the new component. This draws a wire from the Import component to the new component and displays the Add Wire window.

   In the Add Wire window, click **OK**. Another Add Wire window pops up, asking if you want to convert the WSDL interface to Java. Click **No**.

   The new **Standalone Reference** component displays in the Assembly Diagram window with a wire that connects it to the module's Import component.

   Click **File > Save** to save the assembly diagram. For more information, refer to the WebSphere Integration Developer user guide at http://www.ibm.com/software/integration/wid.

Deploying the adapter project

Your project file is a J2EE(TM) Connector project in your workspace in WebSphere(R) Integration Developer. You need to export it to your local file system as an enterprise application archive (EAR) file. Then you need to provide authentication information for JCA connector security before you upload and install the EAR file for the project on the application server.

1. Export the project to an EAR file

   In WebSphere Integration Developer, you need to export your project into an EAR file. On the Export panel in the Select window, choose **EAR file** from the list and click **Next**.

   In the EAR Export window, select the business integration module in the **Ear project** field. The module name **JDBCEMD** now has the suffix **App**. Enter or browse to the location where you want the EAR file to be created. Click all the check boxes to include everything you created in the EAR file. Click **Finish**.

2. Provide authentication information

   Open the WebSphere Process Server administrative console. In the left panel, click **Security > Global security**. In the Global security window, in the Configuration pane, go to the **Authentication** heading on the right side. Click **JAAS Configuration**. Then click the **J2C Authentication data** link.

In the window Global security > J2EE Connector Architecture (J2C) authentication data entries, click **New**. Enter an alias name in the **Alias** field. Enter a user ID and password that can connect to the database. Click **OK**. Click **Save** to save the authentication information.

3. Install the EAR file on the application server

   Using the administrative console, in the left panel click **Applications > Install New Application**. Click **Browse** , select the EAR file, and click **Next**.

   Keep clicking **Next** until you reach **Step 7. Map resource references to resources**. Select the authentication alias you created earlier, click the check box for **Use default method**, and click **Apply** .

   In the following windows, click **Next** or **Continue** until you see the Summary of Installation Options pane. Click **Finish**. You will see the message `Application name installed successfully`.

   Click **Save to Master Configuration** to save your changes. Then in the Enterprise Applications pane, click **Save**.

Related tasks

"Creating a project for the adapter" on page 32
The first task in deploying the adapter is to create a J2EE(TM) connector project for the adapter. The resource adapter archive (RAR) file for the Adapter for JDBC needs to be imported into WebSphere(R) Integration Developer. This sets up the project in your workspace in WebSphere Integration Developer.

Configuring the adapter on the server

Once the project enterprise application archive (EAR) file for the adapter project has been installed on the application server, if you want you can reconfigure properties that the adapter uses to set up a communication channel to a specific database application. Then you can start the configured adapter application.

Normally, the configuration properties are set using the enterprise service discovery wizard when you create your adapter project. You can reset the properties using the WebSphere(R) Process Server administrative console, or you can just note whether the properties are populated according to the values you set in the enterprise service discovery wizard.

1. Open the adapter project

   In the WebSphere Process Server administrative console, in the Configuration pane, the name of your adapter project appears. Under the **Related Items** heading, select **Connector Modules**. The project RAR file name is displayed. Click the check box next to the file name.

   Under **Additional Properties**, select **Resource Adapter**.

2. Reconfigure properties

   a. Edit J2C connection factory property values

      In the administrative console, your adapter project is displayed under **General Properties > Name**. On the right side of the window under **Additional Properties**, click **J2C connection factories**.

      The adapter project name is displayed along with the JNDI name specified in the EJB project. Click the check box next to your adapter's factory connection.

      The J2C connection factory properties and values appear. Edit the values for the properties as needed. For details about these properties see "J2C connection factory properties."

   b. Set values for J2C activation specification properties

If you have *not* already set these values, follow Step 2a to set the values for the J2C activation specification properties, as needed.

Important: If you already specified the J2C activation specification properties using the enterprise service discovery wizard, you cannot change the property settings now. After you deploy the project as a new application on the application server, you cannot update the J2C activation specification properties using the WebSphere Process Server administrative console. If you try to update the settings, the adapter will not recognize the updated values. If you did not set the J2C activation specification properties before installing the project on the application server, you can set these property values now.

To set these properties, under **Additional Properties** click **J2C activation specifications**. For details about these properties see "J2C activation specification properties."

c. Edit the Resource adapter property values

Under **Additional Properties**, click **Custom properties**. A list of the Resource adapter properties and their values is displayed. Edit the property values as needed. See the properties and their descriptions in "Resource adapter properties."

Note: In the **Custom properties** list, at the adapter level, find the **DatabaseVendor** property. This property is required for starting the application. Enter a value for this property if it is blank, and save it.

3. Start the adapter application

In the administrative console, in the panel on the left side, click **Applications > Enterprise Applications**. In the **Enterprise Applications** pane, click the check box next to your application's name and click **Start**. A message will indicate that the application started successfully. If problems occur when you try to start the application, check the adapter log file for error descriptions.

For information on specific messages, see Messages in the IBM WebSphere Adapters Information Center. For details on logging and tracing, see "Troubleshooting."

Related tasks

"Troubleshooting" on page 51
You can contact IBM(R) Software Support when you need to report a problem, following the instructions in this section. You can determine the status of event processing by enabling logging for the adapter. You can set tracing levels to determine the level of errors or warnings captured in the adapter log file. The instructions in this section describe how to enable the Common Event Infrastructure for your adapter. And this section describes mainframe data access.

Related reference

"J2C connection factory properties" on page 72
J2C connection factory properties are used to configure a target enterprise information system (EIS) instance. These properties affect outbound processing and correspond to the ManagedConnectionFactory interface of the J2EE(TM) Connector Architecture Specification.

"J2C activation specification properties" on page 74
J2C activation specification properties activate message endpoints. These properties correspond to the ActivationSpec Interface of the J2EE(TM) Connector Architecture Specification.

"Resource adapter properties" on page 68
These configuration properties are defined at the resource adapter level.

Troubleshooting

You can contact IBM(R) Software Support when you need to report a problem, following the instructions in this section. You can determine the status of event processing by enabling logging for the adapter. You can set tracing levels to determine the level of errors or warnings captured in the adapter log file. The instructions in this section describe how to enable the Common Event Infrastructure for your adapter. And this section describes mainframe data access.

Related tasks

"Configuring the adapter on the server" on page 49
Once the project enterprise application archive (EAR) file for the adapter project has been installed on the application server, if you want you can reconfigure properties that the adapter uses to set up a communication channel to a specific database application. Then you can start the configured adapter application.

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(R) Adapter for JDBC 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. Because 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.

The adapter communicates with the database through a JDBC driver, and the SQL queries or stored procedure calls could result in an SQLException message. These are captured and logged and a ResourceException message is produced. Messages

are returned based on errors occurring during request processing or errors in the Event Store. Errors in event processing are handled by the Event Manager.

Note: When you create and bind both an inbound and outbound operation to the same adapter instance with the same AdapterID property, you need to use the same log name, because only one log file will be created once you deploy the adapter to WebSphere Process Server. If the inbound and outbound operations belong to different adapters, each with a different AdapterID, then a separate log file will be created for each operation.

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

1. Start logging and tracing

   Start the WebSphere Process Server administrative console. From the administrative console, select **Troubleshooting → Logs and Trace**.

2. Specify log detail level

   Click either **Component** to specify a log detail level for individual components, or click **Groups** to specify a log detail for a predefined group of components.

3. Select the logging level

   The ″Logging levels″ table describes the different logging levels that you can set through the administrative console.

   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(R) Service log (when enabled), or the Diagnostic Trace Service (when enabled).

   Logging levels

Level	Indicator	Description
Fatal	F	Task cannot continue. Component cannot function.
Severe	E	Task cannot continue. Component can still function. This also includes conditions that indicate an impending fatal error, that is, 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.
Audit	A	Significant event affecting server state or resources.
Info	I	General information outlining overall task progress.
Config	C	Configuration change or status.

Level	Indicator	Description
Detail	D	General information detailing subtask progress.

4. Save logging information

   Click **Apply** to save your changes.

   Related tasks

   "Setting connection properties" on page 40
   After you have created the adapter project, you need to initialize the enterprise service discovery wizard for the Adapter for JDBC and set the values of the connection properties for your database instance.

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.

You can configure the tracing levels through the WebSphere(R) Process Server administrative console. Follow the steps below to enable and set tracing levels.

1. Start tracing

   Start the WebSphere Process Server administrative console. Select **Troubleshooting → Logs and Trace**.

2. Select the tracing level

   The ″Tracing levels″ table describes the different tracing levels that you can set through the administrative console.

   Tracing levels

Level	Indicator	Description
Fine	1	General trace. Includes broad actions being taken by the adapter, such as establishing a connection to the enterprise information system (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.

3. Save tracing information

   Click **Apply** to save your changes.

   Related tasks

   "Setting connection properties" on page 40
   After you have created the adapter project, you need to initialize the enterprise service discovery wizard for the Adapter for JDBC and set the values of the connection properties for your database instance.

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

Mainframe data access

The Adapter for JDBC supports mainframe data access through use of the IBM(R) WebSphere(R) Information Integrator Classic Federation for z/OS(R). This product provides Web and distributed applications with read/write connectivity to mainframe databases.

The product delivers high-performance SQL-driven access and federation of mainframe data sources. You can use the desktop with your choice of Internet tools and applications to access mainframe, mission-critical information transparently.

If you have problems using the Adapter for JDBC with IBM WebSphere Information Integrator Classic Federation on the IBM z/OS platform, refer to technotes describing configuration requirements on the IBM product support Web pages at http://www-306.ibm.com/software/integration/wbiadapters/support.

Using the example files and sample application

Examples of the following artifacts from the enterprise service discovery process are provided: Service Component Architecture (SCA) input and output files and a corresponding Web Services Description Language (WSDL) file. In addition, a sample application is provided to help you learn how to use the adapter to communicate between J2EE(TM) applications and enterprise information systems.

Example import, export, and WSDL files

Examples are provided for Service Component Architecture (SCA) import and export files and a corresponding Web Services Description Language (WSDL) file. These artifacts are produced during the enterprise service discovery process.

Example export (inbound) file

```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:Export xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:_="http://JDBCEMD/inbound"
xmlns:eis="http://www.ibm.com/xmlns/prod/websphere/scdl/eis/6.0.0"
xmlns:scdl="http://www.ibm.com/xmlns/prod/websphere/scdl/6.0.0"
xmlns:wsdl="http://www.ibm.com/xmlns/prod/websphere/scdl/wsdl/6.0.0"
name="inbound/JDBCInboundInterface">
  <interfaces>
    <interface xsi:type="wsdl:WSDLPortType" portType="_:JDBCInboundInterface"/>
  </interfaces>
  <esbBinding xsi:type="eis:EISExportBinding"
dataBindingType="com.ibm.j2ca.extension.emd.runtime.WBIDataBindingImpl">
    <resourceAdapter name="JDBCEMDApp.IBM JDBC Adapter"
type="com.ibm.j2ca.jdbc.JDBCResourceAdapter">
      <properties/>
    </resourceAdapter>
    <connection type="com.ibm.j2ca.jdbc.inbound.JDBCActivationSpec"
selectorType="com.ibm.j2ca.extension.emd.runtime.WBIFunctionSelectorImpl">
      <properties>
        <BONamespace>http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc</BONamespace>
        <jdbcDriverClass>COM.ibm.db2.jdbc.app.DB2Driver</jdbcDriverClass>
        <databaseURL>jdbc:db2:somedb<databaseURL>
        <password>abcdefg</password>
        <userName>db2admin</userName>
      </properties>
    </connection>
    <methodBinding method="createDb2adminCustomer"
nativeMethod="emitCreateAfterImageDb2adminCustomer"/>
    <methodBinding method="updateDb2adminCustomer"
nativeMethod="emitUpdateAfterImageDb2adminCustomer"/>
    <methodBinding method="deleteDb2adminCustomer"
nativeMethod="emitDeleteAfterImageDb2adminCustomer"/>
  </esbBinding>
</scdl:Export>
```

Example import (outbound) service description

```
 <?xml version="1.0" encoding="UTF-8"?>
<scdl:Import xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:_="http://JDBCEMD/outbound"
xmlns:eis="http://www.ibm.com/xmlns/prod/websphere/scdl/eis/6.0.0"
xmlns:scdl="http://www.ibm.com/xmlns/prod/websphere/scdl/6.0.0"
xmlns:wsdl="http://www.ibm.com/xmlns/prod/websphere/scdl/wsdl/6.0.0"
name="outbound/JDBCOutboundInterface">
  <interfaces>
    <interface xsi:type="wsdl:WSDLPortType" portType="_:JDBCOutboundInterface"/>
  </interfaces>
  <esbBinding xsi:type="eis:EISImportBinding"
dataBindingType="com.ibm.j2ca.extension.emd.runtime.WBIDataBindingImpl">
    <resourceAdapter name="JDBCEMDApp.IBM JDBC Adapter"
type="com.ibm.j2ca.jdbc.JDBCResourceAdapter">
      <properties/>
    </resourceAdapter>
    <connection type="com.ibm.j2ca.jdbc.JDBCManagedConnectionFactory"
interactionType="com.ibm.j2ca.jdbc.JDBCInteractionSpec">
      <properties>
        <databaseURL>jdbc:db2:somedb</databaseURL>
        <jdbcDriverClass>COM.ibm.db2.jdbc.app.DB2Driver</jdbcDriverClass>
        <password>abcdefg</password>
        <userName>db2admin</userName>
      </properties>
    </connection>
    <methodBinding method="createDb2adminCustomer">
      <interaction>
        <properties>
          <functionName>Create</functionName>
        </properties>
      </interaction>
    </methodBinding>
    <methodBinding method="updateDb2adminCustomer">
      <interaction>
        <properties>
          <functionName>Update</functionName>
        </properties>
      </interaction>
    </methodBinding>
    <methodBinding method="deleteDb2adminCustomer">
      <interaction>
        <properties>
          <functionName>Delete</functionName>
        </properties>
      </interaction>
    </methodBinding>
    <methodBinding method="retrieveDb2adminCustomer">
      <interaction>
        <properties>
          <functionName>Retrieve</functionName>
        </properties>
      </interaction>
    </methodBinding>
    <methodBinding method="retrieveallDb2adminCustomer">
      <interaction>
        <properties>
          <functionName>RetrieveAll</functionName>
        </properties>
      </interaction>
    </methodBinding>
    <methodBinding method="applychangesDb2adminCustomer">
      <interaction>
        <properties>
          <functionName>ApplyChanges</functionName>
        </properties>
```

```
      </interaction>
    </methodBinding>
  </esbBinding>
</scdl:Import>
```

Example WSDL file

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="CUSTOMER"
        targetNamespace="http://test/j2c/jdbc/customer"
        xmlns:tns="http://test/j2c/jdbc/customer"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:datans="http://test/j2c/jdbc/customer">
        <wsdl:types>
                <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
                        <xsd:import namespace="http://test/j2c/jdbc/customer"
                                schemaLocation="CUSTOMER.xsd">
                        </xsd:import>
                </xsd:schema>
        </wsdl:types>
        <wsdl:message name="CUSTOMERRequest">
                <wsdl:part name="request" element="datans:CUSTOMER"></wsdl:part>
        </wsdl:message>
        <wsdl:portType name="Customer">
                <wsdl:operation name="updateCustomer">
                        <wsdl:input message="tns:CustomerRequest"></wsdl:input>
                        <wsdl:output message="tns:CustomerRequest"></wsdl:output>
                </wsdl:operation>
                <wsdl:operation name="createCustomer">
                        <wsdl:input message="tns:CustomerRequest"></wsdl:input>
                        <wsdl:output message="tns:CustomerRequest"></wsdl:output>
                </wsdl:operation>
                <wsdl:operation name="retrieveCustomer">
                        <wsdl:input message="tns:CustomerRequest"></wsdl:input>
                        <wsdl:output message="tns:CustomerRequest"></wsdl:output>
                </wsdl:operation>
        </wsdl:portType>
</wsdl:definitions>
```

Sample: Updating a database application

Sample files are provided with the IBM(R) WebSphere(R) Adapter for JDBC so that you can use them to practice updating specific tables in a database application. Two step-by-step scenarios are provided; one is targeted to the application integrator and the other is for the data integrator. Depending on your role, you can practice generating business objects, deploying the adapter, and configuring the adapter to communicate between J2EE(TM) applications and enterprise information systems.

Sample application scenarios

Scenario	Description	Audience
Scenario 1	• Provides already generated artifacts and illustrates how the adapter processes business objects. This scenario does not require use of the enterprise service discovery wizard to generate artifacts. • Intended for those who are 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 wizard to discover database components and to develop the business objects that the adapter processes. • Intended for those who have the same responsibilities as the application integrator, but are also responsible for enabling access to a range of data sources for the application developers.	Data Integrator

For more details on the roles of application integrator and data integrator, see the "Audience" topic in the "Product Overview."

Related concepts

"Audience" on page 2
The information in this topic defines the users of the WebSphere Adapter products and details the skills they require.

Obtaining the project file for Scenario 1

For Scenario 1, the sample includes a comprehensive set of files so you do not need to generate them. In an actual work environment, someone would need to generate the artifacts and configure the adapter using the enterprise service discovery wizard.

Before using the scenarios, you need to install the adapter and extract the samples package from the folder called "Sample."

For information about installing the adapter, refer to Installing IBM WebSphere Adapters.

Obtain the project file

Go to the "jdbc/samples/Apps" folder and find the JDBCApp.ear.

The enterprise application archive (EAR) file for this project contains the following files:

- A configured instance of the adapter that is deployed by default to the local host application server: CWYBC_JDBC.rar
- A service component architecture (SCA) module with various SCA artifacts:
  - JDBCInboundInterface.export
  - JDBCInboundInterface.wsdl
  - JDBCOutboundInterface.import

  - JDBCOutboundInterface.wsdl
- A WSDL file: CustomerInterface.wsdl
- Business objects:
  - InboundRtasserCustomer.xsd
  - InboundRtasserAddress.xsd
  - InboundRtasserCustInfo.xsd
  - OutboundRtasserCustomer.xsd
  - OutboundRtasserAddress.xsd
  - OutboundRtasserCustInfo.xsd
  - InboundRtasserCustomerBG.xsd
  - InboundRtasserAddressBG.xsd
  - InboundRtasserCustInfoBG.xsd
  - OutboundRtasserCustomerBG.xsd
  - OutboundRtasserAddressBG.xsd
  - OutboundRtasserCustInfoBG.xsd
  - InboundRtasserCustomerContainer.xsd
  - InboundRtasserAddressContainer.xsd
  - InboundRtasserCustInfoContainer.xsd
  - OutboundRtasserCustomerContainer.xsd
  - OutboundRtasserAddressContainer.xsd
  - OutboundRtasserCustInfoContainer.xsd
- SQL scripts to create database tables:
  - CreateCustomerTable.sql
  - CreateAddressTable.sql
  - CreateCustInfoTable.sql

Next, you will do the deployment and configuration task.

Obtaining the project file for Scenario 2

For Scenario 2, the sample includes a minimal set of files so that you can practice generating a comprehensive set of artifacts using the enterprise service discovery wizard.

Before using one of the scenarios, you need to install the adapter and extract the samples package from the folder called “Sample.”

For information about installing the adapter, refer to Installing IBM WebSphere Adapters.

Obtain the project file

Go to the “jdbc/samples/Apps” folder and find the JDBCApp.ear.

The enterprise application archive (EAR) file for this project contains the files listed below. These files are examples of the artifacts you will create in this scenario when you use the enterprise service discovery wizard to generate artifacts and configure the adapter for deployment and use. When you complete the scenario, you can view the files to compare them with your output. The files include sample import, export, WSDL and business object files, as follows:

- JDBCOutboundInterface.import

- JDBCInbound Interface.export
- JDBCOutboundInterface.wsdl
- JDBCInboundInterface.wsdl
- InboundRtasserCustomer.xsd
- InboundRtasserAddress.xsd
- InboundRtasserCustInfo.xsd
- OutboundRtasserCustomer.xsd
- OutboundRtasserAddress.xsd
- OutboundRtasser.CustInfo.xsd
- InboundRtasserCustomerBG.xsd
- InboundRtasserAddressBG.xsd
- InboundRtasserCustInfoBG.xsd
- OutboundRtasserCustomerBG.xsd
- OutboundRtasserAddressBG.xsd
- OutboundRtasserCustInfoBG.xsd
- InboundRtasserCustomerContainer.xsd
- InboundRtasserAddressContainer.xsd
- InboundRtasserCustInfoContainer.xsd
- OutboundRtasserCustomerContainer.xsd
- OutboundRtasserAddressContainer.xsd
- OutboundrtasserCustInfoContainer.xsd

For information about business objects, see "Business objects overview."

Next you will do the deployment and configuration task.

Deploying and configuring for Scenario 1

For Scenario 1, the sample provides all of the artifacts so that you don't need to use the enterprise service discovery wizard to create them.

The sample provides an instance of the adapter that is already configured. In the following procedure, you need to update the business object application-specific information (ASI) to match your environment. And you need to change the values of the configuration properties to match the database instance in your enterprise information system (EIS).

1. Start WebSphere(R) Integration Developer

   For details, refer to the WebSphere Integration Developer user guide at http://www.ibm.com/software/integration/wid.
2. Create the business integration module

   In WebSphere Integration Developer, go to the Business Integration perspective. Right click in the Business Integration pane. Click **New > Module**. In the New Module window, enter **JDBC** for the **Module Name**. Click **Finish**.
3. Import the Resource Adapter Archive (RAR) file
   a. Click **File > Import**. In the Select window, select **RAR file** as the import source and click **Next**. This imports CWYBC_JDBC.rar into WebSphere Integration Developer.
   b. In the Connector Import window, at the **Connector file** field, browse to the path and name of the module created in the previous step, for example,

install dir \jdbc\deploy\CWYBC_JDBC.rar. Ensure that the check box for **Add module to an EAR project** is checked. In the **EAR project** field, select **JDBCApp**. Click **Finish**.

c. In the Business Integration perspective, in the Business Integration pane, right-click on the **JDBC** project and select **Open Dependency Editor**. Click the arrow to expand the **J2EE** section. Click **Add**. In the J2EE(TM) Project Selection window, select **CWBC_JDBC** and click **OK**.From the menu, click **File > Save**.

4. Add JDBC driver to Java build path

   You need to add a reference to the JDBC driver to your project.

   In WebSphere Integration Developer at the Java perspective, right click the **CWBC_JDBC Connector Project**. Select **Properties**.

   To add the external JAR file, click **Java Build Path**. Select the **Libraries** tab and click **Add External Jars**. In the File System window, navigate to the **JDBC Driver** and select the JAR file.

   Click **OK** to save your changes.

5. Add business object artifacts to the Business Integration module

   You need to uncompress the JDBCApp.ear.

   Next you need to extract the JAR file from the enterprise application archive (EAR) file. From JDBCAPP.ear, extract JDBC.jar to \workspace\JDBC.

6. Change the business object application-specific information

   a. At the Business Integration perspective, right click the **JDBC** module and select **Refresh**.

   b. Click the **+** next to **Data Types**. Right click the business object **InboundRtasserCustomer.xsd**.

   c. Select **Open With > Business Object Editor**. Select the **Properties** tab. Select the **Application Info** tab.

   d. In the ASI Properties pane, change the value of jdbcasi:TableName from **RTASSER.CUSTOMER** to *schema***.CUSTOMER** using the name of the schema in your database where the CUSTOMER table has been created. From the menu, click **File > Save**.

   Repeat steps 6a through 6d for the following business objects:

   - InboundRtasserAddress
   - InboundRtasserCustInfo
   - OutboundRtasserCustomer
   - OutboundRtasserAddress
   - OutboundRtasserCustInfo

7. Provide authentication information

   a. You need to create an authentication alias, using the administrative console of WebSphere Process Server. Open the administrative console. In the left panel, click **Security > Global security**. In the Global security window, in the Configuration pane, go to the **Authentication** heading on the right side. Click **JAAS Configuration**. Then click the **J2C Authentication data** link.

   b. In the window Global security > J2EE Connector Architecture (J2C) authentication data entries, click **New**. Enter an alias name in the **Alias** field. Enter a user ID and password that can connect to the database. Click **OK**. Then click **Save** to save the authentication information.

8. Change values for configuration properties

   a. You need to change the configuration property values to match your database instance. In WebSphere Integration Developer at the Business

Integration perspective in the Business Integration pane, click **JDBC module**. Double-click **Inbound/JDBCInboundInterface**.

b. In the assembly editor, at the **Properties** tab, click the **Binding** tab, then click the **Connection** tab.

c. Expand **ActivationSpecProperties**. Set the properties UserName, Password, DatabaseURL, and JDBCDriverClass to the appropriate values for the J2C activation specification, based on your database configuration.

d. In the **Connection** tab, expand **Authentication Properties**. In the **J2C Authentication Data Entry** field, enter the name of the authentication alias you created in the previous step. From the menu, click **File > Save**.

You need to repeat this procedure for the Outbound/JDBCOutboundInterface object. Update the values for the J2C connection factory (also called Managed Connection Factory) properties: UserName, Password, DatabaseURL, and JDBCDriverClass.

9. Deploy the project file to the application server

You need to export the EAR file for your project from WebSphere Integration Developer to WebSphere Process Server.

a. Export the EAR file

From the WebSphere Integration Developer menu, click **File > Import**. Select the EAR file and click **Next**. In the EAR Export window, go to the **EAR project** field and enter **JDBCApp**. Set the destination path and filename. Select all three of the check boxes and click **Finish**.

b. Install the EAR file on the application server

Using the administrative console, select **Applications > Install New Application**. Click **Browse**, select the EAR file, and click **Next**.

Keep clicking **Next** until you reach **Step 7. Map resource references to resources**. Select the authentication alias you created earlier, select the check box and click **Apply** .

On the following screens, click **Next** or **Continue** until you see the Summary of Installation Options window. Then click **Finish**.

You will see the message **Application SampleApp installed successfully**. Click **Save to Master Configuration** to save your changes. Then click **Save**.

Next you can run the sample application.

Deploying and configuring for Scenario 2

In Scenario 2, you will set adapter connection properties and generate business objects. You will export your project to an enterprise application archive (EAR) file, deploy the project on the application server, and reset configuration properties.

1. Start WebSphere(R) Integration Developer

For details, refer to the WebSphere Integration Developer user guide at http://www.ibm.com/software/integration/wid.

2. Create the business integration module

In WebSphere Integration Developer, go to the Business Integration perspective. Right click in the Business Integration pane. Select **New > Module**. In the New Module window, enter `JDBC` for the **Module Name**. Click **Finish**.

3. Import the adapter RAR file

In WebSphere Integration Developer, select **File > Import** to import the adapter RAR file. The RAR file is jdbc/deploy/CWYBC_JDBC.rar.

In the Select window, select **RAR file** as the import source.

In the Connector Import window, at the **Connector file** field, browse to the path and name of the module created in the previous step, for example, *install dir*/jdbc/deploy/CWYBC_JDBC.rar.

Ensure that the check box for **Add module to an EAR project** is selected. In the **EAR project** field, select **JDBCApp**. Click **Finish**.

4. Add the JDBC driver to the Java build path

   You need to add a reference to the JDBC driver to your project.

   In WebSphere Integration Developer at the **Java** perspective, right-click the **CWBC_JDBC Connector Project**. Select **Properties**.

   To add the external JAR file, click **Java Build Path** . Select the **Libraries** tab and click **Add External Jars**. In the File System window, navigate to the **JDBC Driver** and select the JAR file.

   Click **OK** to save your changes.

5. Provide authentication information

   Open the WebSphere Process Server administrative console. In the left panel, select **Security > Global security**. In the Global security window, in the Configuration pane, go to the **Authentication** heading on the right side. Click **JAAS Configuration**. Then click the **J2C Authentication data** link.

   In the window Global security > J2EE Connector Architecture (J2C) authentication data entries, click **New**. Type an alias name in the **Alias** field. Enter a user ID and password that can connect to the database. Click **OK**. Click **Save** to save the authentication information.

6. Set adapter connection properties

   a. In WebSphere Integration Developer, at the **Business Integration** tab that contains the JDBC connector project, right click on the frame. In the pop-up menu select **New > Enterprise Service Discovery**.

   b. In the Select an Enterprise Service Resource Adapter window, select **JDBC EMD Adapter** and click **Next**.

   c. In the Configure Settings for Discovery Agent window, enter the information for your database in the connection configuration property fields. Only four connection property settings are required for the sample. For details on these properties, see "Connection properties" in the "Reference" section. The following table shows example values for initializing the discovery agent.

      Metadata discovery connection properties

Property	Example Values
UserName	db2admin
Password	xxxxxxxxxxx
DatabaseURL	jdbc:db2:mytestdb
JDBCDriverClass	COM.ibm.db2.jdbc.app.DB2Driver

   Click **Next**.

7. Query for database objects.

   After configuring the connection properties, you can run a query for database objects. You can browse the metadata tree structure to understand the structure of objects in the enterprise information system (EIS), and make selections of objects needed for the service description.

At this point you could specify Filter Properties to narrow the list of schemas displayed in the tree structure. See the procedure for specifying filter properties in "Querying for database objects."

In the Find and Discover Enterprise Services window, click **Run Query**.

The metadata objects are displayed in the top pane. Highlight an object to generate from and click **Add** . The selected object appears in the bottom pane. You can remove any selected object by clicking **Remove**.

When you are finished selecting objects, click **Next**.

8. Specify Selection Properties.

   After you have selected database objects, you need to specify values for the Selection properties for the import and export files.

   The NameSpace property is initially set to a default value for all business objects.

   Selecting the Service Type (inbound or outbound) and corresponding operations are described in "Setting Selection properties." After specifying these selection properties, click **Next**.

9. Save the adapter project

   The module name **JDBC** automatically appears in the Generate Artifacts window. Do not change the check box setting. Click **New** to create a new Business Integration module where the artifacts will be saved.

   In the New Module window, the module name appears. A default folder name is selected. This is where the Service Descriptions will be saved.

   In the **J2C Authentication Data Entry** field, type the name of the authentication alias you created earlier.

   Click **Finish**.

   You can compare your output to the sample files. In the Business Integration perspective at the Business Integration pane, click **+** next to the **JDBC** module to expand it. Click **+** next to the **Data Types** folder to view a list of all the business objects.

10. Deploy the project file to the application server

    Your project file is a J2EE Connector project in your workspace in WebSphere Integration Developer. You need to export it to your local file system. Then, you need to provide authentication information for JCA connector security before you upload and install the EAR file on the application server.

    a. Export the EAR file

       From the WebSphere Integration Developer Export screen in the Select window, select the EAR file **JDBCApp** and click **Next**.

       In the EAR Export window, you need to select the business integration module and the location where the EAR file should be created. Select the module in the **EAR project** field. Enter or browse to the location where you want the EAR file to be created. Click all the check boxes and click **Finish**.

    b. Install the EAR file on the application server

       Using the administrative console, on the left pane select **Applications > Install New Application**. Click **Browse**, select the EAR file, and click **Next**.

       Keep clicking **Next** until you reach **Step 7. Map resource references to resources**. Select the authentication alias you created earlier, select the check box for **Use default method**, and click **Apply** .

       On the following screens, click **Next** or **Continue** until you see the Summary of Installation Options window. Then click **Finish**.

You will see the message **Application SampleApp installed successfully**. Click **Save to Master Configuration** to save your changes. Then in the Enterprise Applications pane, click **Save**.

11. Change values for configuration properties

    The adapter properties for UserName, Password, DatabaseURL and JDBCDriverClass are already set to default values, but you must reconfigure them to match your database instance, using the administrative console.

    For inbound processing, you need to modify the J2C activation specification properties. For outbound processing, you need to change the J2C connection factory properties. The sample does not require that you reconfigure values for any additional properties. You can also change Resource adapter configuration properties to the settings you prefer. For details on these properties, see the "Configuration" properties in the "Reference" section.

    a. Open the adapter project file

       In the administrative console, at the **Configuration** tab of the Enterprise Applications window, click the name of your adapter project. Under the **Related Items** heading, select **Connector Modules**. The project RAR filename is displayed. Click the check box next to the file name.

       At the **Configuration** tab, under the **Additional Properties** heading, select **Resource Adapter**.

    b. Reconfigure properties

       Your adapter project is displayed under **Name** in the General Properties pane. Under **Additional Properties**, click one of the property categories--either J2C activation specification, J2C connection factory, or Custom properties--to edit the property values. Custom properties are the Resource adapter properties.

Next you can run the sample application.

Related tasks

"Querying for database objects" on page 42
After configuring the connection properties, you can run a query for database objects. You can browse the metadata tree structure to understand the structure of objects in the enterprise information system (EIS), and make selections of objects needed for the service description.

"Setting Selection Properties" on page 45
After you have selected database objects, you need to specify values for the Selection properties for the import and export files.

Related reference

"Connection properties" on page 77
Connection properties are used by the enterprise service discovery wizard for connecting to the target enterprise information system (EIS) instance.

"Configuration properties" on page 68
This group of properties contains attributes used by the adapter to set up a communication channel to a specific database application.

Running the sample application

In this task, you will run the adapter sample application to test the processing of an event during inbound operations. You can check the accuracy of the output values from the event.

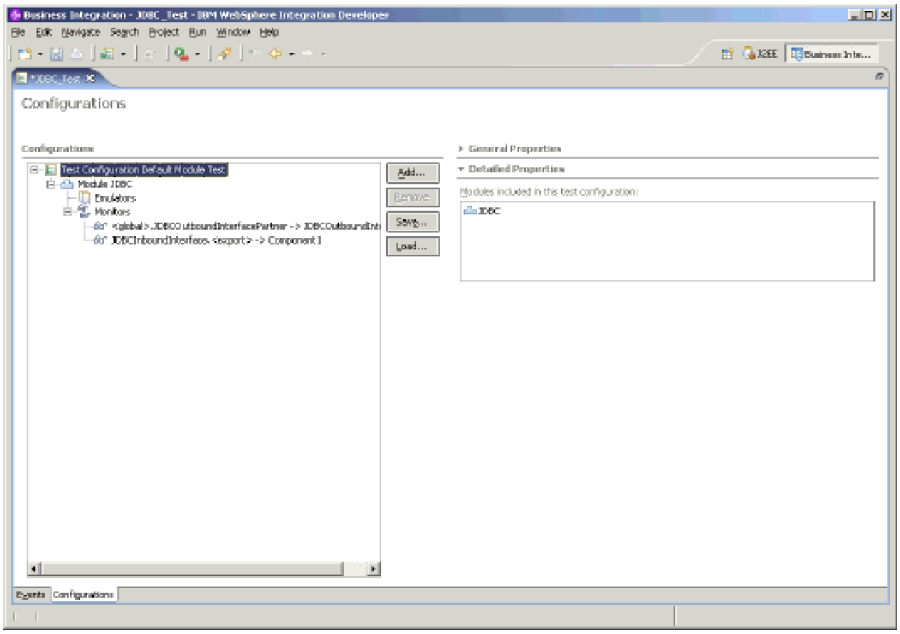
You will run the sample using the WebSphere(R) Integration Developer test client.

1. Select the Business Integration module and server

From the Business Integration perspective of WebSphere Integration Developer, select the **JDBC** module in the Business Integration tab. Right-click **Test > Attach**.

The WebSphere Integration Developer test client displays the Events window. On the right side, **Default Module Test** is displayed in the **Configuration** field. The **Module** field shows **JDBC**.

Click the **Configurations** tab on the bottom left. From the Configurations window, click **+** beside **Monitors** to ensure that the monitor **JDBCInboundInterface <export> Component1** exists for the export. If you completed Scenario 1, this monitor file was generated for you. If you completed Scenario 2, this file was created in Step 9, "Save the adapter project. "



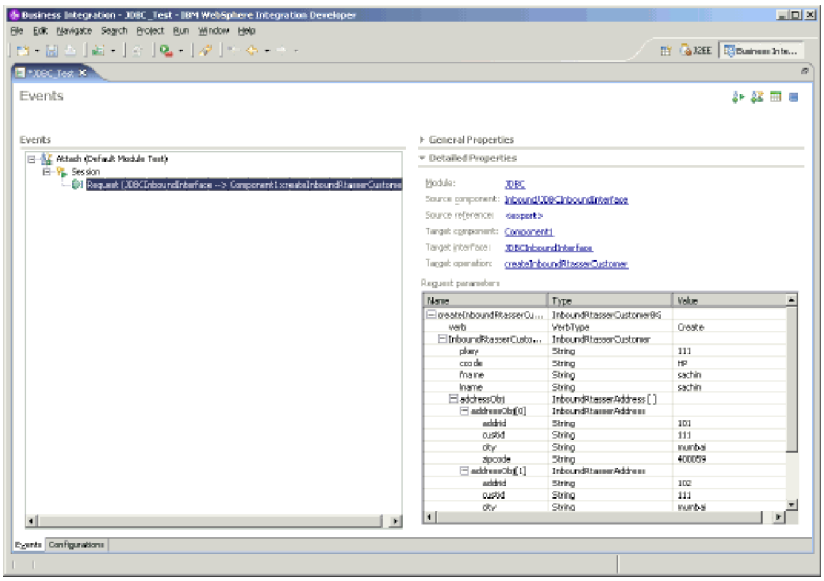
Click the **Events** tab on the bottom left side and then click **Continue**. From the Select Deployment Location pop-up window, select the application server to run the application. Select **WebSphere Process Server v6.0** and click **Finish**. The pop-up window **Starting the integration test client** appears.

2. Test event processing

   To test event processing, you can insert an event manually into the sample application event table. Use the following example as a guide for inserting the event into the event store with a SQL statement:

   ```
   INSERT INTO wbia_jdbc_eventstore (object_key, object_name, object_function,
   event_priority, event_status)
   VALUES ('11234', 'InboundRtasserCustomerBG', 'Create', 1, 0)
   ```

   When the monitored component receives the event, an entry appears in the Events window. See the following example Events window.



Reference

The section describes the properties for configuration, connection, and object selection.

Related concepts

"Transaction management" on page 13
The Adapter for JDBC supports both local and XA transactions.

Configuration properties

This group of properties contains attributes used by the adapter to set up a communication channel to a specific database application.

The types of properties are:

- Resource adapter
- J2C connection factory
- J2C activation specification

Bidirectional properties are described for each of these property types.

Related tasks

"Saving the adapter project" on page 46
After you specify selection properties, you configure properties that the adapter uses to set up a communication channel to a specific database. These include Resource adapter, J2C connection factory, J2C activation specification, and bidirectional properties. You also need to create a new business integration module where all the artifacts and property values can be saved.

"Deploying and configuring for Scenario 2" on page 63
In Scenario 2, you will set adapter connection properties and generate business objects. You will export your project to an enterprise application archive (EAR) file, deploy the project on the application server, and reset configuration properties.

Resource adapter properties

These configuration properties are defined at the resource adapter level.

The table “Resource adapter properties” describes these configuration properties. The bidirectional resource adapter properties that need to be configured when bidirectional text transformation has been activated are described in a separate table in this section.

Resource adapter properties

Property	Type	Globalized	Description
BONameSpace	String	No	Namespace for the business object definitions to be used by this adapter. This value should be taken from the value you provided during the metadata discovery process. This property is required.
DatabaseVendor	String	No	Specifies which RDBMS the adapter uses for special processing. Set the value to `DB2`, `Oracle`, or `MSSQLServer` if using an IBM DB2, Oracle, or Microsoft SQL Server database. If using a different database, set the value to the name of that database. If using a non-default database, ensure that the proper driver is loaded. If this property is set to `Others`, the adapter determines which database to use by locating the driver. A value is required for the adapter to process successfully.
LogFileName	String	No	The full path of the log file. This property is required.
LogNumberOfFiles	Integer	No	The number of log files to use. When a log file reaches its maximum size, another log file is started. If no value is specified, the value is set to 1.

Property	Type	Globalized	Description
LogMaxFileSize	Integer	No	Size of the log files in kilobytes. If no value is specified, the files have no maximum size.
PingQuery	String	No	SQL query used to test valid connection to the database.
QueryTimeOut	Integer	Yes	This specifies the maximum number of seconds a query can take for all SQL statements. If a value is not specified, no timeout is set on the query. If the query takes longer than the number of seconds specified, the database produces an SQL exception that is captured. The associated message is logged in the log file.
ReturnDummyBO ForSP	Boolean	No	Used to return output parameters even when the result set is empty. In the case of RetrieveSP, a result set is returned. If the result set is empty, no business objects are created, and the output parameters returned by the procedure call cannot be retrieved. If this property value is `true`, a dummy business object with values from output and input/output parameters populated in the corresponding attributes is returned. The default value is `false`.
TraceFileName	String	No	The full path to the trace file. This property is required.

Property	Type	Globalized	Description
TraceNumberOfFiles	Integer	No	The number of trace files to use. When a trace file reaches its maximum size, another trace file is started. If no value is specified, the value is set to 1.
TraceFileSizeMax	Integer	No	Size of the trace files in kilobytes. If no value is specified, the files have no maximum size.

The following table describes Resource adapter properties that need to be configured only when bidirectional text transformation has been activated.

Bidirectional resource adapter properties

Property	Type	Description
BiDiContextEIS	String	Defines bidirectional format for content data in all of the business object runtime instances.
BiDiContextMetadata	String	Defines bidirectional format for metadata or configuration properties stored in the deployment descriptor.
BiDiContextSkip	Boolean	Flag that controls the invocation of bidirectional support at the resource adapter level. If it equals `true`, the support will not be invoked. If it equals `false`, then the support will be invoked.
BiDiContextSpecialFormat	String	Specifies the category of properties subject to special treatment for all of the connector properties.
BiDiContextTurnBiDiOff	Boolean	Flag that controls the invocation of bidirectional transformation for the JDBC adapter. This parameter takes precedence over all the rest of the bidirectional parameters on all levels. In other words, if it is set to `true`, no bidirectional support is invoked, and no checks nor lookups are performed. If it is set to `false`, bidirectional support is invoked.

Related tasks

"Configuring the adapter on the server" on page 49
Once the project enterprise application archive (EAR) file for the adapter project has been installed on the application server, if you want you can reconfigure properties that the adapter uses to set up a communication channel to a specific database application. Then you can start the configured adapter application.

J2C connection factory properties

J2C connection factory properties are used to configure a target enterprise information system (EIS) instance. These properties affect outbound processing and correspond to the ManagedConnectionFactory interface of the J2EE(TM) Connector Architecture Specification.

A J2C connection factory manages connection pooling. It provides configuration information for outbound connectivity to a single JDBC application instance from an application by way of the adapter.

The table titled "J2C connection factory properties" defines the JDBC configuration properties that pertain to a J2C connection factory. Two additional tables in this section define the bidirectional J2C connection factory properties that need to be configured only when bidirectional text transformation has been activated.

J2C connection factory properties

Property	Type	Globalized	Description
AutoCommit	Boolean	No	Value for AutoCommit to be set on the connection.
DatabaseURL	String	Yes	Database URL used to connect to the database. This is enabled for use with bidirectional languages.
JDBCDriverClass	String	No	JDBC driver class for the driver that is used to connect to the database.
Password	String	Yes	Password for the corresponding user name. This is enabled for use with bidirectional languages.
UserName	String	Yes	User name to log in to the enterprise information system. This is enabled for use with bidirectional languages.
XADataSourceName	String	No	XA datasource name to establish an XA connection to the database.

Property	Type	Globalized	Description
XADatabaseName	String	Yes	Database name used for the XA connection. This property must be set if you are using a DB2 database.

Bidirectional J2C connection factory properties

Property	Type	Description
BiDiContextEIS	String	Defines bidirectional format for content data in all of the business object runtime instances supported by the adapter for specific connection to the EIS.
BiDiContextMetadata	String	Defines bidirectional format for metadata or configuration properties for specific connection to the EIS.
BiDiContextSkip	Boolean	Flag that controls the invocation of bidirectional support at the resource adapter level. If it equals `true`, the support will not be invoked. If it equals `false`, then the support will be invoked.
BiDiContextSpecialFormat	String	Specifies the category of properties subject to special treatment for all of the connector properties of the specific connection to the EIS.
DatabaseURL BiDi properties		*DatabaseURLEIS, DatabaseURLSpecialFormat, DatabaseURLSkip
Password BiDi properties		*PasswordEIS, PasswordSpecialFormat, PasswordSkip
UserName BiDi properties		*UserNameEIS, UserNameSpecialFormat, UserNameSkip

*The three bidirectional (BiDi) properties associated with each BiDi-supported property. These properties are described in the following table.

BiDi property	Type	Description
BiDiContextCP<property_Name>EIS	String	Defines bidirectional format for a specific connector configuration property for a specific connection to the EIS.

BiDi property	Type	Description
BiDiContextCP<property_ Name>SpecialFormat	String	Flag that controls the invocation of bidirectional support at the connector configuration property level. If it equals `true`, the support will not be invoked. It if equals `false`, then the support will be invoked.
BiDiContextCP<property_ Name>Skip	Boolean	Defines the special format for a specific connector property for a specific connection to the EIS.

Related concepts

"Transaction management" on page 13
The Adapter for JDBC supports both local and XA transactions.

Related tasks

"Configuring the adapter on the server" on page 49
Once the project enterprise application archive (EAR) file for the adapter project has been installed on the application server, if you want you can reconfigure properties that the adapter uses to set up a communication channel to a specific database application. Then you can start the configured adapter application.

J2C activation specification properties

J2C activation specification properties activate message endpoints. These properties correspond to the ActivationSpec Interface of the J2EE(TM) Connector Architecture Specification.

The table titled, "J2C activation specification properties" describes the JDBC adapter-specific configuration properties that pertain to activating message endpoints. A separate table in this section describes the bidirectional J2C activation specification properties that need to be configured only if bidirectional text transformation has been activated.

J2C activation specification properties

Property	Type	Globalized	Description
AutoCreateEDT	Boolean	Yes	Flag that indicates whether the adapter should create the EDT table automatically if it does not already exist. The default value is `true`.
DatabaseURL	String	Yes	The driver-specific URL for creating a connection to the enterprise information system (EIS). This is enabled for use with bidirectional languages.
Delivery Type	String	Yes	This property determines the order in which the events are published. The value is either `ORDERED` or `UNORDERED`.*Ordered* means events are published one at a time, while *Unordered* means events are published all at once. The default value is `ORDERED`.
EDTDatabaseName	String	Yes	The name of the database for the event distribution table.

Property	Type	Globalized	Description
EDTDriverName	String	Yes	Name of the XA database driver to use to connect to the event distribution table for inbound events, for example: `com.ibm.db2j.DB2jXADataSource` If no value is present, the event manager cannot perform recovery.
EDTSchemaName	String	Yes	The schema name for the event distribution table.
EDTTableName	String	Yes	The name of the table that represents the event distribution table.
EDTUserName	String	Yes	The user name required to connect to the database for the event distribution table.
EDTUserPassword	String	Yes	The password required to connect to the database for the event distribution table.
EventOrderBy	String	No	The order in which events are retrieved and processed. Expected values are comma-separated column names of the event table. This property is enabled for use with bidirectional languages.
EventTableName	String	Yes	Table in the EIS that contains events generated by the EIS for inbound processing.
JDBCDriverClass	String	No	JDBC driver class used to connect to the EIS.
Password	String	Yes	Password for authorizing the user to retrieve events from the EIS. This property is enabled for use with bidirectional languages.
PollPeriod	Integer that is equal to or greater than 0	Yes	The rate in milliseconds at which to poll the EIS event store for new inbound events. If `0`, the adapter will not wait between cycles. The poll cycle is established at a fixed rate, meaning that if an execution of the poll cycle is delayed (for example, the prior poll cycle takes longer than expected to complete) the next cycle will occur immediately to catch up. This is a required property. The default value is `500`.
PollQuantity	Integer that is greater than 0.	Yes	This property is used to determine the number of events to deliver to each endpoint per poll cycle. It is a required property.
Username	String	Yes	User name for logging into the EIS for inbound events. This is enabled for use with bidirectional languages.

Bidirectional J2C activation specification properties

Property	Type	Description
BiDiContextEIS	String	Defines bidirectional format for content data in all of the business object runtime instances supported by the adapter for specific connection to the EIS.
BiDiContextMetadata	String	Defines the bidirectional format for metadata or configuration properties for a specific connection to the EIS.
BiDiContextSkip	Boolean	Flag that controls the invocation of bidirectional support at the resource adapter level. If it equals `true`, the support will not be invoked. If it equals `false`, then the support will be invoked.
BiDiContextSpecialFormat	String	Specifies the category of properties subject to special treatment for all of the connector properties of a specific connection to the EIS.
DatabaseURL BiDi properties		*DatabaseURLEIS, DatabaseURLSpecialFormat, DatabaseURLSkip
Password BiDi properties		*PasswordEIS, PasswordSpecialFormat, PasswordSkip
UserName BiDi properties		*UserNameEIS, UserNameSpecialFormat, UserNameSkip
EventTableName BiDi properties		*EventTableNameEIS, EventTableNameSpecial Format, EventTableNameSkip
EventOrderBy		*EventOrderByEIS, EventOrderBySpecialFormat, EventOrderBySkip
EDTTableName		EDT_BiDiFormat, EDT_BiDiSkip, EDTURL_BiDiSpecialFormat, EDTURL_BiDiSkip
EDTUserName		EDT_BiDiFormat, EDT_BiDiSkip, EDTURL_BiDiSpecialFormat, EDTURL_BiDiSkip

Property	Type	Description
EDTUserPassword		EDT_BiDiFormat, EDT_BiDiSkip, EDTURL_BiDiSpecialFormat, EDTURL_BiDiSkip
EDTDatabaseName		EDT_BiDiFormat, EDT_BiDiSkip, EDTURL_BiDiSpecialFormat, EDTURL_BiDiSkip

*The three bidirectional (BiDi) properties associated with each BiDi-supported property. These properties are described in the following table.

BiDi property	Type	Description
BiDiContextCP<property_ Name>EIS	String	Defines bidirectional format for a specific connector configuration property for a specific connection to the EIS.
BiDiContectCP<property_ Name>SpecialFormat	String	Flag that controls the invocation of bidirectional support at the connector configuration property level. If it equals `true`, the support will not be invoked. If it equals `false`, then the support will be invoked.
BiDiContextCP<property_ Name>Skip	Boolean	Defines special format for a specific connector property for a specific connection to the EIS.

Related tasks

"Configuring the adapter on the server" on page 49
Once the project enterprise application archive (EAR) file for the adapter project has been installed on the application server, if you want you can reconfigure properties that the adapter uses to set up a communication channel to a specific database application. Then you can start the configured adapter application.

Connection properties

Connection properties are used by the enterprise service discovery wizard for connecting to the target enterprise information system (EIS) instance.

Related tasks

"Deploying and configuring for Scenario 2" on page 63
In Scenario 2, you will set adapter connection properties and generate business objects. You will export your project to an enterprise application archive (EAR) file, deploy the project on the application server, and reset configuration properties.

Metadata discovery connection properties

The enterprise service discovery process requires these properties to connect to the enterprise information system (EIS) for discovery and for creating the service description.

Using these properties, the discovery service prepares a metadata tree to be displayed for object selection and navigation.

The table titled "Metadata discovery connection properties" describes these properties.

Metadata discovery connection properties

Property	Required	Globalized	Description
Database URL	Yes	Yes	The database URL to connect to. The URL may vary from driver to driver. Specify this property according to the driver instructions. This property is enabled for use with bidirectional languages.
JDBCDriverClass	Yes	No	The name of the database driver used to connect to the database.
Password	Yes	Yes	Password for the corresponding user name. This property is enabled for use with bidirectional languages.
Prefix	No	Yes	The prefix to be added to the name of the business object. This property is enabled for use with bidirectional languages.
UserName	Yes	Yes	User name for logging in to the database. This property is enabled for use with bidirectional languages.

Related tasks

"Setting connection properties" on page 40
After you have created the adapter project, you need to initialize the enterprise service discovery wizard for the Adapter for JDBC and set the values of the connection properties for your database instance.

Bidirectional connection properties

These properties enable the enterprise service discovery wizard to apply the proper bidirectional transformation on the data passed to the enterprise information system (EIS).

The following table lists the bidirectional properties that are used during connection configuration.

Bidirectional connection properties

Property	Value	Letter Position	Description
BiDi OrderingSchema	Implicit Visual	1	Logical (Implicit). Implicit is the default. Visual
BiDi Direction	LTR RTL contextual_LTR contextual_RTL	2	Left to Right. LTR (L) is the default. Right to Left Contextual Left to Right Contextual Right to Left
BiDi SymmetricSwapping	Yes No	3	Symmetric swapping is on. Yes (Y) is the default. Symmetric swapping is off
BiDi Shaping	nominal shaped initial middle final isolated	4	Nominal (N). This is the default. Letter shaped according to its position. Letter is in its initial shape. Letter is in its middle shape. Letter is in its final shape. Letter is in its isolated shape.
BiDi NumericShaping	nominal national contextual	5	Nominal (N). This is the default. Hindi (National) Contextual

Related tasks

"Setting connection properties" on page 40
After you have created the adapter project, you need to initialize the enterprise service discovery wizard for the Adapter for JDBC and set the values of the connection properties for your database instance.

Object selection properties

These properties are used during the business object selection process. They include Filter, Node, and Selection properties.

Filter and Node properties

During business object discovery, you can specify Filter properties if you want to narrow the list of schemas displayed in the tree; otherwise, all schemas are displayed. You can set the Node property if you want to narrow the list of database objects to display.

The table titled "Filter properties" describes the properties you need to use to filter the schemas.

Filter properties

Filter property	Type	Description
SchemaNameFilter	String	String used to filter schemas. Only those schemas that start with the string you specify are displayed. If this property is not set, all schemas are displayed.

Filter property	Type	Description
Types	Multi-value	Lists the database object types: tables, views, stored procedures, and synonyms/ nicknames. Values are set by enterprise service discovery. Multiple selections are allowed.
DefineASI	Boolean	If `true`, when you select an object for generation, you are prompted to set the application-specific information (ASI) values for the business object.

The table titled "Node property" describes the property you need to use to filter database objects.

Node property

Node property	Type	Description
ObjectNameFilter	String	String used to filter database objects. Only those database objects that start with the text you specify are displayed. If no ObjectNameFilter is specified, all objects are displayed.

Related concepts

"Discovery of system capabilities" on page 34
The adapter discovers business objects in a database by analyzing the database to identify the schemas. Then it creates a list of all the objects from the database and shows it as a tree structure.

Selection properties

After you have selected database objects, you need to set values for the Selection properties.

The enterprise service discovery wizard queries for the Selection properties. The table titled "Selection properties" describes these properties.

Selection properties

Selection property	Type	Description
BOLocation	String	The path to the location where the generated .xsd files are to be stored. Will be set as RelativePath on DataDescription.
MaxRecords	Integer	The maximum number of records to retrieve for a RetrieveALL operation.
NameSpace	String	Text entry which is used to filter schemas.

Selection property	Type	Description
Operations	Multi-value	Defaults to a list of the operations supported by the adapter for the selected ServiceType. Select from these operations. The specified operations are set for all business objects being generated.
ServiceType	String	Either inbound or outbound is chosen. Values are set by enterprise service discovery.

The NameSpace property is initially set to the default NameSpace for all business objects. The default namespace is `http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc`.

The NameSpace is prepended to the business object name to keep business object schemas logically separated. For example: `http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/Schema1Customer`.

Related concepts

"Object selection and generation" on page 35
To generate business objects, you select database object nodes. Then the enterprise service discovery wizard generates business objects for the objects of the selected nodes.

Related tasks

"Setting Selection Properties" on page 45
After you have selected database objects, you need to specify values for the Selection properties for the import and export files.

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