WebSphere® Adapters

Version 6 Release 1

WebSphere Adapter for JDBC User Guide
Version 6 Release 1

WebSphere® Adapters

Version 6 Release 1

WebSphere Adapter for JDBC User Guide
Version 6 Release 1

Note
Before using this information and the product it supports, read the information in "Notices" on page 227.

16 January 2007

This edition applies to version 6, release 1, modification 0 of IBM WebSphere Adapter for JDBC and to all subsequent releases and modifications until otherwise indicated in new editions.

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

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

© Copyright International Business Machines Corporation 2006, 2007. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2006, 2007

Chapter 1. Overview of WebSphere Adapter for JDBC

With WebSphere Adapter for JDBC, you can create integrated applications that include the exchange of information with a database. By using the adapter, an application can send requests to the database, as well as receive events from the database, often without the need for SQL code.

The adapter enables two-way communication between an application running on WebSphere Process Server or WebSphere Enterprise Service Bus and a database. Using the adapter, an application can send requests to read, create, modify, or delete data in a database, in many cases without writing any SQL code. To process requests received from an application, the adapter updates the database tables using SQL queries or stored procedures. An application can also receive events from the database, for example, it can be notified that specific database tables are updated. To process events that result from changes to the database, the adapter delivers events to an application. Using event notification, updates to the database can be automatically propagated to other applications. By combining event processing by WebSphere Adapter for JDBC and another adapter, updates can be automatically propagated to enterprise applications such as Siebel, PeopleSoft, and Oracle.

The adapter provides a standard interface that integrates with diverse database software vendors and versions; it supports any database server with a Java™ Database Connectivity (JDBC) driver that supports the JDBC 2.0 or later specification. Examples of such servers include IBM® DB2®, Oracle, Microsoft® SQL
2 Server, Sybase, Derby, and Informix®. The adapter uses business objects to exchange data between the application and the database, so the application does not need to use the JDBC application programming interface (API). *Business objects* are containers for application data that represent business functions or elements, such as a database table or the result of an SQL query. The adapter understands the data format provided by the application, and can process the data, perform the operation, and send the results back in that format.

New in this release

WebSphere Adapter for JDBC, Version 6.1.0 provides enhancements to the adapter. This release also deprecates some features.

Updates to this information are made available at the WebSphere Adapters product support Web site. To read updated or additional information, see: http://www.ibm.com/software/integration/wbiadapters/support/.

Deprecated in version 6.1.0

A deprecated feature is one that is supported but no longer recommended and that might become obsolete. For a list of features from earlier versions of Adapter for JDBC that have been deprecated in version 6.1.0, see "Deprecated features" on page 54.

New in version 6.1.0

- Support for batch SQL scripts (user-specified SQL INSERT, UPDATE, and DELETE statements)

You can now create business objects for batch SQL scripts that run a series of INSERT, UPDATE, and DELETE SQL statements against the database.

- Change to the timing of object retrieval for events

  The database object that is the subject of the event is not retrieved until after notification is delivered to the export. As a result, the detection and notification of any retrieval errors that occur is deferred until after notification of the export. This differs from the event processing in version 6.0.2 of the adapter, where retrieval errors can be detected before the adapter notifies the export.
- Business graphs and verbs are now optional

  The business graph that contains each business object in version 6.0.2 is now optional. You need a business graph only for modules whose business objects were created in version 6.0.2 or for new version 6.1.0 modules that use the ApplyChanges outbound operation.
- Changes to the enterprise service discovery wizard

  The wizard has been renamed the external service wizard, and includes functional and usability improvements to make it easier for you to discover, create, and configure business objects and services for use with the adapter. The wizard now guides you through several tasks that were previously performed manually in the file system or in WebSphere Integration Developer, such as creating a project, importing the JDBC driver into the project, and creating the module.

  The wizard now provides default values for many properties, makes it easier to enter certain information, indicates which properties are required, and lets you configure the module without worrying about advanced properties.
- Simplified support for bidirectional script processing
- Support for node-level, or stand-alone, deployment of the adapter
- Support for business faults

  The adapter now generates business faults for business exceptions. This lets you easily assign a corrective action for those error conditions.
- The adapter RAR file is available in WebSphere Integration Developer; you do not need to install it separately. The wizard automatically copies the adapter files into the project for you.
- The adapter documentation is located on the WebSphere Integration Developer Information Center, in the Configuring and using adapters section.
- Support for a first-failure data capture (FFDC) construct that can be contained in a WebSphere Application Server symptom database to provide information and suggested actions to assist a diagnostic module in customizing the data that is logged.
- Support for IPv6 addresses

Introduced in version 6.0.2, fix pack 2

- Connection properties in addition to user name and password

  Use the new DriverConnectionProperties property in the activation specification (for inbound processing) and managed connection factory (for outbound processing) to specify additional connection properties.
- Support for data sources that cache prepared statements

  Improve the performance of inbound and outbound processing using caching of prepared statements if a data source is used to connect to the database.
- Improved display of stored procedure names in the external service wizard

  Easily locate a stored procedure in the external service wizard because the stored procedure name is displayed in a more readable way.

- Improved support for business object hierarchies
  - Use the external service wizard instead of the business object editor to build a hierarchy of child business objects that sets foreign keys appropriately in parent and child business objects
  - Group unrelated business objects in a business object hierarchy by using the external service wizard to create wrapper business objects
- Support for globalized data types in Oracle and MS SQL databases

  You no longer need to use the business object editor to manually add attributes for columns with globalized data types in Oracle and MS SQL. The data types NCHAR, NVARCHAR, NTEXT, TEXT, RAW, MONEY, and SMALLMONEY are now supported.
- Support for NULL values in Update, Delete, Retrieve, and RetrieveAll operations

  The adapter can update or retrieve a record from a database table when the column value is null.
- Additional quick start tutorials

  Learn about the adapter using additional quick start tutorials, which are available on the Web.

Introduced in version 6.0.2

- Support for the SQL data types CLOB (character large object) and BLOB (binary large object).
- Support for stored functions. Stored functions are similar to stored procedures, except that they always return a value. The adapter checks for ReturnValue application-specific information and, if it exists, the adapter runs the stored function.
- Custom event processing: Custom queries use either standard SQL, a stored procedure, or a stored function.
- EDT is no longer used for assured once delivery. A new Activation specification property, AssuredOnceDelivery, is provided for this function.
- A new filter can be used to narrow the list of stored procedures in enterprise metadata discovery when associating a stored procedure with a business object
- The adapter can filter events to be processed by business object type using the Activation specification property EventFilterType, and it can filter events by timestamp using the property FilterFutureEvents.
- The ability to generate query business objects from user-specified SELECT statements during enterprise service discovery.
- The Execute operation to support stored procedures and stored functions.
- Support for establishing database connections through a data source defined in WebSphere Application Server. A DataSourceJNDIName property has been added to the J2C managed connection factory and activation specification properties.
- High availability support for inbound processing. For more information, see “WebSphere Adapters in clustered environments” on page 51.

Hardware and software requirements

The hardware and software requirements for WebSphere Adapters are documented on the IBM Web site at the location below.

Hardware and software requirements for WebSphere Adapters:
http://www.ibm.com/support/docview.wss?uid=swg27006249

Additional information

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

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

Technical overview of WebSphere Adapter for JDBC

The adapter supports integration of databases that are accessible through the JDBC application programming interface (API) with applications running on WebSphere Process Server or WebSphere Enterprise Service Bus. The adapter provides outbound and inbound processing under the Java 2 Platform, Enterprise Edition (J2EE) Connector architecture (JCA) and integrates with Service Component Architecture (SCA) components.

Outbound processing enables an application to access or modify data in a database. The adapter converts a request from the application to an outbound operation, which it runs to create, retrieve, update, or delete data in the database or to run a database program stored in the database. Processing these requests results in the creation, retrieval, update, or deletion of rows in the corresponding database tables. The adapter also enables you to run stored procedures or store functions that are defined in the database, and to run user-defined SELECT, INSERT, UPDATE, and DELETE statements. You can use the adapter to integrate multiple applications with the same database.

Figure 1 on page 5 presents an overview of the flow of outbound processing. An application running in WebSphere Process Server or WebSphere Enterprise Service Bus invokes a service in an outbound module, which sends a request to the adapter to process one or more business objects. The adapter uses the JDBC API to connect to the database server, which accesses the tables and other objects in the database.

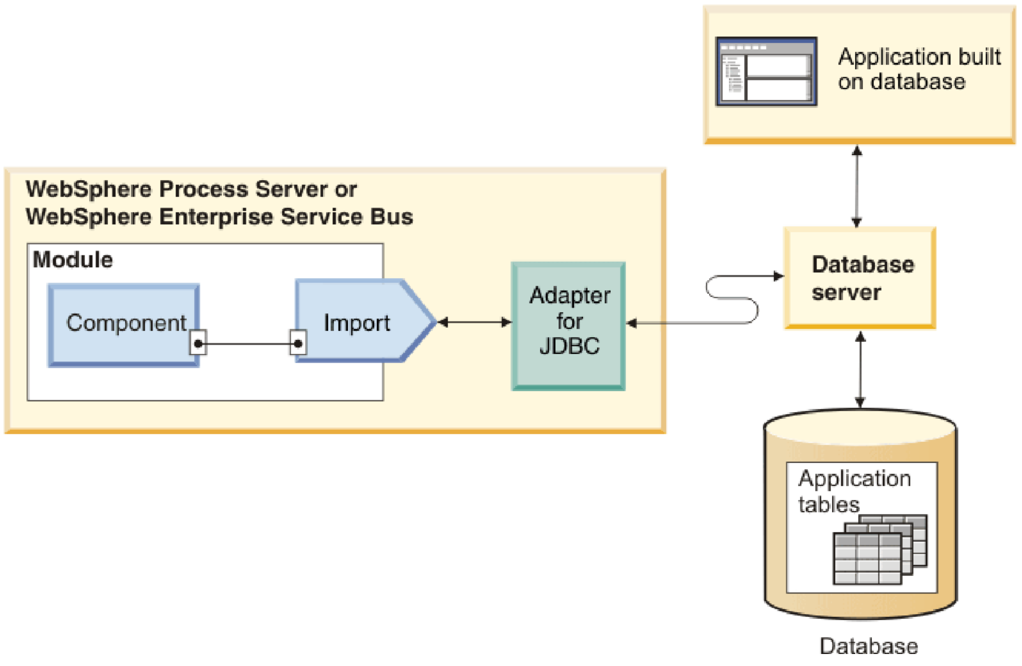

Application built on database
WebSphere Process Server or WebSphere Enterprise Service Bus
Module
Component
Import
Adapter for JDBC
Database server
Application tables
Database

Figure 1. Processing outbound requests

Inbound processing enables an application to receive notification when objects in the database are changed. For example, an application can be notified when rows are created, updated, or deleted in selected database tables.

Figure 2 on page 6 presents an overview of the flow of inbound processing. A database application changes tables in the database. The change causes a trigger, or another automated mechanism, to update the event store with information about the change. Periodically, the adapter polls the event store, retrieves and processes events, and then delivers them to the export of a module that is part of an application that runs in WebSphere Process Server or WebSphere Enterprise Service Bus.

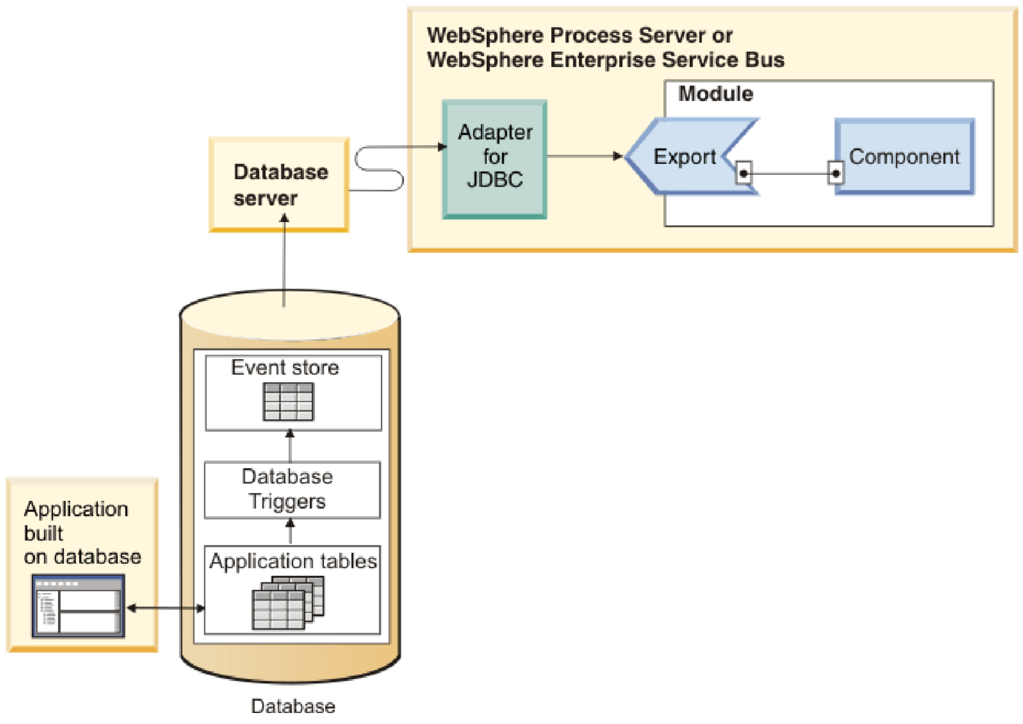


Figure 2. Processing inbound events

The adapter can process events in one of the following ways:

- Standard event processing, using an event store that is populated by the database application
- Custom event processing, using a user-defined database query

During *standard event processing*, when data is changed in the tables in the database, appropriate events are inserted into an database table called an event store, along with relevant information, such as key values. To capture the changed data, you can place triggers on the respective tables, or use other methods such as Oracle Change Data Capture, which is provided for Oracle databases. The adapter polls the event store and retrieves a batch of events. The events can be filtered by business object type and time stamp. The adapter uses each event to construct a business graph or business object that contains the business objects changed by that event. The business object or business graph is then dispatched to the exports that are configured to receive the specific business object.

During *custom event processing*, the adapter runs a query that was specified by the user as a standard SQL statement, a stored procedure, or a stored function. Any of these actions returns a result set for data returned by the query. Each row of the result set corresponds to a row in the event store. The adapter constructs a business object for each event and delivers it to the exports (also called endpoints) that are configured for (or have subscribed to) the specific business object.

For both standard and custom event processing, you can specify how often the adapter polls for events and how many events it retrieves each polling period.

Outbound processing

When an application component needs to retrieve or modify data in the database, the adapter acts as the connector between the application component and the database. The adapter provides a set of standard outbound operations, which process either after-image or delta style business objects. The adapter also supports both local and XA (distributed) transactions for outbound processing.

The adapter business object model uses two styles of business objects for making updates: after-image and delta. An *after-image* business object is one that contains the complete state of the business object after all desired changes have been made to it. A *delta* business object is one that contains only key values and the data to be changed. Delta business objects are used only in operation that update business objects.

Supported operations

Table 1 lists the outbound operations that are supported for each type of business object and indicates whether each supports after-image or delta style processing.

Table 1. Outbound operations supported by type of business objects

2

Business objects supported	**Operation**	**After-image support**	**Delta support**
Tables Views Synonyms Nicknames	Create	Yes	No
	Update	Yes	No
	Delete	Yes	No
	Retrieve	Not applicable	Not applicable
	RetrieveAll	Not applicable	Not applicable
	ApplyChanges	Yes	Yes
Stored procedures Batch SQL	Execute	Not applicable	Not applicable
Queries	RetrieveAll	Not applicable	Not applicable
Wrappers	Create	Yes	No
	Update	Yes	No
	Delete	Yes	No
	Retrieve	Yes	No

Transaction management

The adapter supports both local and XA (distributed) transactions for outbound processing. In the adapter, a transaction is an isolated interaction with the database. 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 users of the database.

The adapter supports transactions only if the database server supports transactions. The types of transactions that are supported are local and XA transactions:

- A *local transaction* is one in which a component defines the start and end of the transaction with a single database. It uses a one-phase commit protocol.
- An *XA transaction* is one in which the transaction can span multiple heterogeneous databases. It uses a global, or two-phase commit, protocol.

XA transactions

The adapter supports XA transactions for outbound processing. Chose one of these methods to configure the adapter for XA transactions:

- Specify a JNDI data source that supports XA transactions, using the DataSourceJNDIName property
- Specify an XA data source and database, using the XADataSourceName and XADatabaseName properties, respectively

The DataSourceJNDIName property represents a data source created within WebSphere Process Server or WebSphere Enterprise Service Bus. This name represents an XA or connection pool data source. If you define a JNDI data source that supports XA transactions on the server and specify that data source when you configure the adapter, the adapter supports any type of database that supports XA transactions. If you use an XA data source and database, the adapter supports XA transaction only for DB2 and Oracle databases.

Outbound operations

Application components use operations to perform actions such as retrieving from a database. The adapter provides certain outbound operations. Details are provided on how the adapter processes business objects for each of the supported operations.

An operation can be performed using a standard SQL statement provided by the adapter or by a stored procedure that you define. You can run a stored procedure to perform the operation or to do custom processing before or after the operation. In each business object, you can configure how each operation is performed.

Create operation:

The Create operation creates rows in database tables corresponding to the business object in the request. When given a hierarchical business object, the Create operation recursively traverses the business object, creating rows corresponding to each business object in the hierarchy.

To process the Create operation, the adapter performs the following actions:

1. Checks whether the business objects is a wrapper. If the top level business object is a wrapper business object, the adapter ignores the business object. No rows are created for wrapper objects.
2. 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.
3. Retrieves and checks for the existence of each single-cardinality child business object contained without ownership. If the retrieval 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 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.
4. Inserts the top-level business object in the database by performing the following actions:

   a. Sets each of the foreign-key values of the top-level business object 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. 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.
   c. Inserts the top-level business object into the database.
5. Processes each of its multiple-cardinality child business objects as follows:
   a. 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. Inserts each of the multiple-cardinality child business objects into the database.

Retrieve operation:

The Retrieve operation extracts data from a database for a hierarchy of business objects.

To process the Retrieve operation, the adapter performs the following actions:
1. Removes all child business objects from the top-level business object it received. In other words, it makes a copy of the top-level business object without any children.
2. Retrieves the top-level business object from the database.
   - If the top level business object is a wrapper business object, it is ignored. No retrieval is performed for wrapper business objects.
   - 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 `RecordNotFoundException` error.
   - If the retrieval returns more than one row, the adapter returns the `MultipleMatchingRecordsException` error.

   The Retrieve operation uses only the primary key. Other columns are ignored.
3. 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. Recursively retrieves each of the single-cardinality children, regardless of 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.

Retrieving NULL data

The adapter can retrieve a record from a database table when the column value is NULL. For example, a Customer business object might have these columns: custid, ccode, fname, and lname, where custid and ccode form composite keys. Composite keys are primary keys that refer to more than one attribute and are used to define the uniqueness of the business object. You can retrieve a Customer record for which ccode is NULL. The adapter generates a SELECT statement for the Retrieve operation as:

```
select custid, ccode, fname, lname from customer where custid=? and ccode is null
```

RetrieveAll operation:

The adapter uses the RetrieveAll operation to retrieve an array of business objects from the database. The process the adapter uses differs depending on whether the RetrieveAll operation is for database table business objects or for user-specified SQL business objects.

For database table business objects

All of the key and non-key attributes populated in the incoming business object determine the selection criteria for the retrieval. 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 name of a generated business object matches the name of the table in the database. For example, the Customer table in the database is represented as a business object named "Customer".

To retrieve an array of business objects, the adapter performs the following actions:

1. Constructs a container for all of the retrieved rows. The name of the container business object is the business object name with the string "Container" appended to it.
2. If the module was configured to use business graphs, which are optional, constructs a top-level business graph for each of the retrieved rows. The name of the business graph is the business object name with the string "BG" appended to it.
3. Retrieves each of the business graphs in the container using the Retrieve operation.

The following figures show the structure of objects returned from a RetrieveAll operation, with and without business graphs.

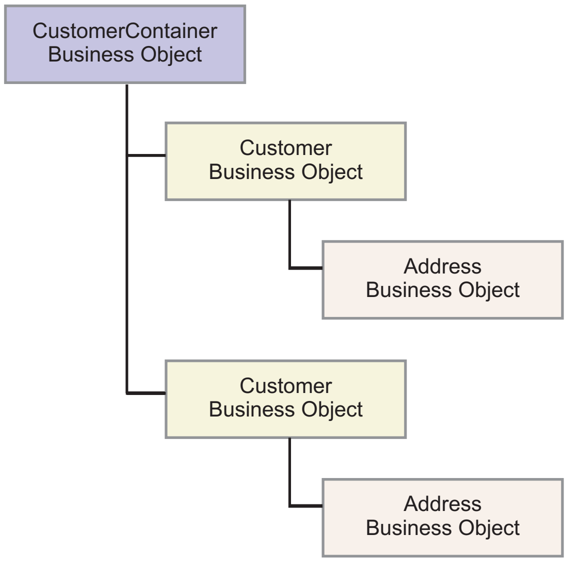


Figure 3. Structure of the business object returned in a RetrieveAll operation without optional business graphs

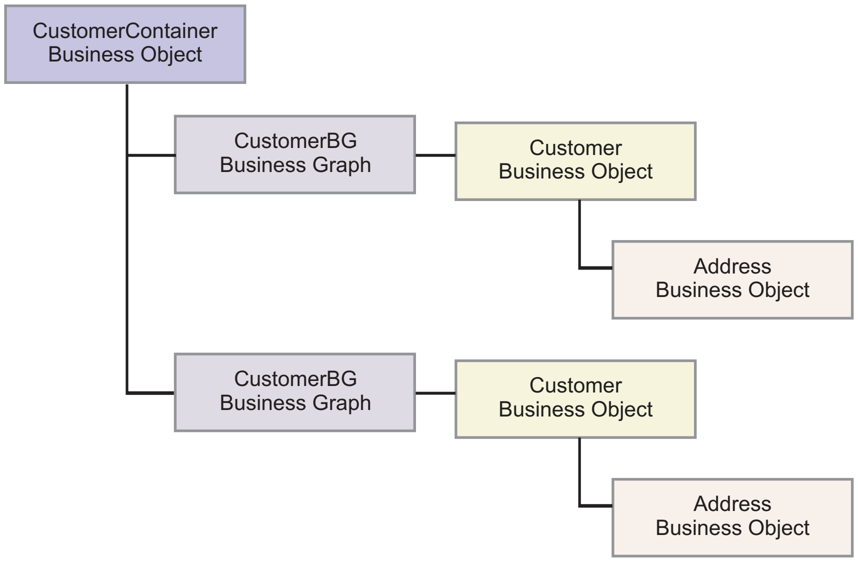


Figure 4. Structure of the business object returned in a RetrieveAll operation with optional business graphs

The following errors can result from a RetrieveAll operation:

- `RecordNotFoundException` – One or more of the populated business objects in the input object does not exist in the enterprise information system.
- `MatchesExceededLimitException` – The number of matching records in the database exceeds the value of the Maximum records for RetrieveAll operation property that is defined in the interaction specification. The MatchCount attribute of the fault contains the actual number of matches that the adapter found in the database, so that you can either increase the limit, or refine the search appropriately.

2 **Note:** If the Maximum records for RetrieveAll operation property is set to a very
2 large number, problems can occur related to a lack of sufficient memory,
2 depending upon the size and number of business objects returned.

- `EISSystemException` – One or more unrecoverable errors are reported by the database (the enterprise information system, or EIS).

For query business objects

Business objects that are created for user-specified SELECT statements (query business objects) also support the RetrieveAll operation. The external service wizard generates the query business object by running the user-specified SQL SELECT statement and creating a hierarchy of query business objects. If you are using optional business graphs, the hierarchy is shown in Figure 5.

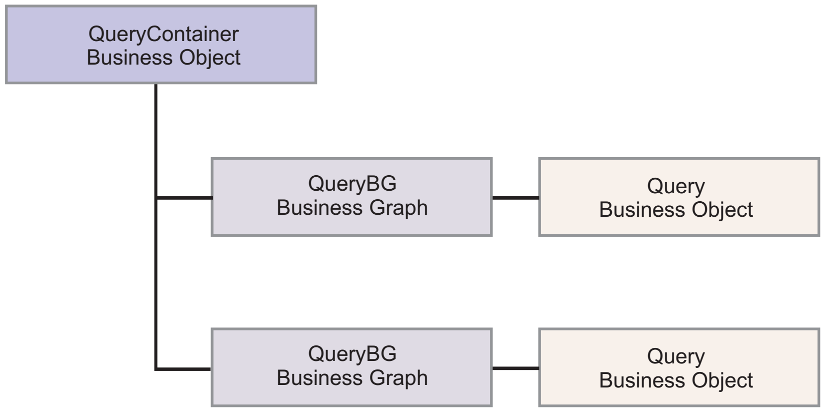


Figure 5. User-specified query business objects

If you are not using optional business graphs, the hierarchy is shown in Figure 6.

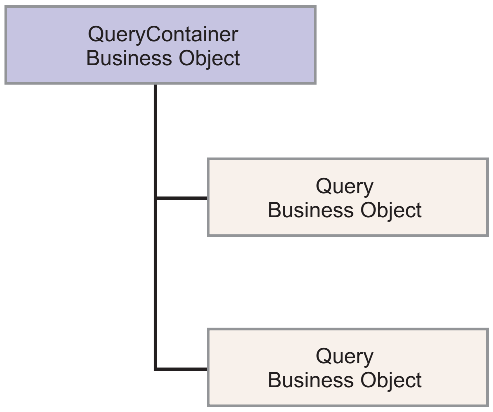


Figure 6. User-specified query business objects

To process the query business object generated by the external service wizard for the user-specified SELECT statement, the adapter performs the following actions:

1. Obtains the SELECT SQL statement from the query business object.
2. Checks whether a dynamic WHERE clause is specified in the query business object.
   - If there is a dynamic WHERE clause, the adapter replaces the default WHERE clause in the SELECT statement with the dynamic one.

- If there is no dynamic WHERE clause, the adapter replaces parameters in the SELECT statement with the corresponding values specified in the query business object.

3. Runs the SELECT statement.
4. Obtains the result set that is returned and populates the query business object values with the data returned from the database, creating a container business object with the structure shown in Figure 5 on page 12.
5. Retrieves the entire hierarchy (a *deep retrieve*) of each top-level query business object in the container, if any child business objects are defined for the query business objects.

   Note: A query business objects can be a top-level business object only. A query business object cannot have child query business objects.

Retrieving NULL objects

The adapter can retrieve records from a database table when the column value is NULL. For example, a Customer business object might have these columns: custid, ccode, fname, and lname, where ccode need not be a primary key. You can query all of the Customer records for which ccode is NULL. The adapter generates a select query for the RetrieveAll operation as:

```
select custid, ccode, fname, lname from customer where custid=? and ccode is NULL
```

Update operation:

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

When updating a hierarchical business object, the adapter performs the following actions:

1. 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.
2. 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 the child in the database.
- If the child is present in the retrieved business object but not in the source business object, the adapter recursively deletes the child 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 the adapter never modifies single-cardinality children contained without ownership.

3. 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.

   If the top level business object is a wrapper business object, it is ignored. No update is done for wrapper business objects.
4. 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 array of the retrieved business object, the adapter recursively creates it in the database.
   - If the child exists in the array of the retrieved business object 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 the KeepRelationship property set to `true`. In this case, the adapter does not delete the child from the database.

NULL data and the Update operation

The adapter can update a record from a database table when the column value is NULL. For example, a Customer business object might have these columns: custid, ccode, fname, and lname, where custid and ccode form composite keys. Composite keys are primary keys that refer to more than one attribute and are used to define the uniqueness of the business object. You can update a Customer record for which ccode is NULL. The adapter would generate an update query for the Update operation as:

```
update customer set fname=?, lname=? where custid=? and ccode is null
```

ApplyChanges operation:

The ApplyChanges operation provides both delta and after-image support for modifying or deleting a business object. The ApplyChanges operation is available only if you use business graphs.

If you set the verb property of the business graph to the name of an operation, such as create, update, or delete, the adapter performs after-image processing for the ApplyChanges operation. For example, if the verb is set to `create`, the adapter processes the ApplyChanges operation the same as the Create operation.

If you do not set the verb in the business graph, the adapter uses the ChangeSummary in the business graph to update the business object. In this mode, the ApplyChanges operation differs from the Update operation in the following ways:

- In the ApplyChanges 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 operation set in the ChangeSummary for each child business object. If a child does not have a operation set in it, the adapter returns an error.

The adapter performs the following steps when updating a hierarchical business object from the ChangeSummary. It processes only the 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 ApplyChanges 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 operation is taken and the appropriate operation is performed. The allowed operations on a child in ApplyChanges are Create, Delete, and Update:

   - If a Create operation 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 operation is found in the child, that child is deleted.
   - If an Update operation is found in the child, the child is 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 application-specific information of the business object. 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.

Physical deletes

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.

  If the top-level business object is a wrapper object, it is ignored. No delete is done for wrapper business objects.
- It recursively deletes all single-cardinality child business objects contained with ownership.

Logical deletes

For logical deletes the adapter takes the following actions:
- It issues an update that sets the status attribute of the business object to the value specified by the business object-level 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.

NULL data and the Delete operation

NULL data and the Delete operation The adapter can delete a record from a database table when the column value is null. For example, a Customer business object might have these columns: custid, ccode, fname, and lname, where custid and ccode form composite keys. Composite keys are primary keys that refer to more than one attribute and are used to define the uniqueness of the business object. You can delete a Customer record for which ccode is null. The adapter generates a delete query for the Delete operation as:

```
delete from customer where custid=? and ccode is null
```

Execute operation:

The Execute operation is used to run stored procedures and stored functions. The external service wizard generates the required stored procedure business object that corresponds to the stored procedure or stored function definition in the database. The adapter uses the Execute operation to process the stored procedure business object.

The following information provides a simple example of a stored procedure, the business object that is constructed from it, and the steps the adapter uses to process the stored procedure business object.

A simple example of a stored procedure:

```
PROCEDURE testSP(IN int x,INOUT VARCHAR(10) msgSTR, OUT int status,
                OUT struct outrec, OUT array retArr)
```

The procedure returns two result sets.

For this stored procedure, an example of the business object that is constructed:

```
BOLevel ASI
      SPName=testSP
      ResultSet=true
      MaxNumberOfResultSets=2
      ReturnValue = propName
                    Returned if the stored procedure is a function. function).
                    Will be property name corresponding to the child business
                    object if returned value is complex type(array/struct/resultset)
          Defined only if it is a Function

Properties
      x Type=IP
      msgStr Type=IO
      status Type=OP
      outrec Type OP - Child BO for outrec, ASI ChildBOType = struct
      retarr Type OP - n cardinality child BO for retArr, ASI ChildBOType = array
      childBOName1 - Child BO for 1st result set, ASI ChildBOType = resultset
      childBOName2 - Child BO for 2nd result set, ASI ChildBOType = resultset
```

To process this stored procedure business object with an Execute operation, the adapter:

1. Constructs the following stored procedure call: `CALL testSP(x, msgStr, status, outrec)`.
2. Sets the input parameters `x` and `msgStr` on the callable statement.
3. Runs the callable statement.
4. Obtains the return value (if Function) and sets the value in the appropriate attribute if it is a scalar value, or in a child business object if it is a complex value (such as struct, array).
5. Obtains the first result set and creates the container for ResultSet1.
6. Obtains the second result set and creates the container for ResultSet2.
7. Obtains the output parameters `msgStr` and `status`, and sets the corresponding attributes on the business object.
8. Obtains the output parameter `outrec` and creates the child business object from the data returned in `outrec`. If `outrec` is a nested struct type, then the adapter recursively creates and stores data in the hierarchical child business object.
9. Obtains the output parameter `retArr` and creates a multiple cardinality child business object from the data returned in `retArr`. If `retArr` is a nested array type, then the adapter recursively creates and stores data in the hierarchical child business object.

Inbound processing

The adapter supports inbound event management with event delivery. Events are processed from an event store that is populated either by the database application or from the result of custom queries that you provide. You control how often the adapter polls for events and how many records are delivered to the export at one time.

The adapter polls for changes using one of these methods:

- Standard event processing, in which the adapter examines the event store for events that are stored there by the database application
- Custom event processing, in which the adapter runs user-defined queries, stored procedures, or stored functions

You can customize standard or custom event processing when you use the external service wizard to configure the adapter initially or at a later time by using the administrative console of the server to change the activation specification properties.

The database object that is the subject of the event is not retrieved until after notification is delivered to the export. As a result, the detection and notification of any retrieval errors that occur is deferred until after notification of the export. This differs from the event processing in version 6.0.2 of the adapter, where retrieval errors can be detected before the adapter notifies the export.

Standard event processing

In standard event processing, the adapter provides the SQL queries that poll for events and ensure that the event is delivered exactly one time.

Database triggers or tools such as Oracle Change Data Capture run when records are created, updated, or deleted in tables in the database. A trigger or other tool writes an event record into the *event store*, which is a persistent cache where event records are saved until a polling adapter can process them. The event store is implemented as a table in same database as the user tables, which are the tables that contain the database objects accessed by the adapter.

You must define the triggers or set up other tools to report changes to the database tables about which you want to receive events. The adapter provides a sample database script that shows how to set up triggers for the adapter. The samples are located in the *WID_installation_dir*/ResourceAdapters/JDBC_*version*/samples/ scripts directory, where *version* identifies the version of the adapter, for example, `6.1.0.0_IF1`. A sample script is provided for IBM DB2, IBM DB2 for z/OS®, Oracle, and Microsoft SQL Server.

The adapter offers assured once delivery, which guarantees that each event is delivered once and only once to the export. If you enable assured once delivery for the module, a transaction ID (XID) is set for each event in the event store. After an event is obtained for processing, the XID value for that event is updated in the event store. The event is then delivered to its corresponding export, and subsequently deleted from the event store. If the database connection is broken or the application is stopped before the event can be delivered, the event cannot be processed completely. In this case, the XID column indicates that the event must be reprocessed and sent to the export again. After the database connection is re-established or the adapter starts again, the adapter checks for events in the event store that have a value in the XID column. The adapter processes these events first, and then polls the other events during the poll cycles.

The adapter can process all events or filter events by business object type. The filter is set by use of the activation specification property EventFilterType. This property has a comma-delimited list of business object types. Only the types specified in the property are processed. If no value is specified for the property, no filter is applied, and all the events are processed. If the activation specification property FilterFutureEvents is set to true, the adapter filters events by timestamp. The adapter compares the system time in each poll cycle to the time stamp on each event. If an event is set to occur in the future, it will not be processed until that time.

Custom event processing

2 In custom event processing, you provide the SQL queries or stored procedures that
2 poll for events.

With custom event processing, you control which events are delivered to the export by providing a database query (the *custom event query*) for the adapter to run in place of the SQL query it uses to poll the event store in standard event processing. The custom event query must perform any necessary filtering. You specify that you want custom event processing by selecting an option in the wizard or by setting the EventQueryType activation specification property in the administrative console.

Custom event processing supports assured once delivery if you create the standard event store for storing XID values. The adapter stores the events returned by the custom event query in the event store, and it updates the events with XID values. The adapter processes the events in the same way as for standard event processing. Do not create a custom query that queries the standard event store, because that table temporarily holds the events when the adapter is configured for assured once delivery. In addition, in this situation the event store must not have an automatic generation of event ID values, because the adapter populates the event ID value it retrieves from the custom query in the event store.

You turn custom event processing on by selecting an advanced option in the wizard when you configure your module to use the adapter or by setting the EventQueryType activation specification property.

Custom event query

You specify the custom event query to run by providing a user-defined event query in an advanced option in the wizard or by setting the CustomEventQuery activation specification property. Specify one of the following types of programs:
- Standard SQL statements
- A stored procedure
- A stored function

Any of these programs takes an input parameter containing the poll quantity, an activation specification property that the adapter provides at run time. The program can accept other input parameters as well. These programs must return a result set that has the poll quantity number of records and contains the following columns in order: event_id, object_key, object_name, and object_function. The adapter generates the event object from the result set and processes the events.

Standard SQL statements

You can provide an SQL SELECT statement that selects the events to process. The query can have input parameters in addition to the input parameter for poll quantity.

Stored procedure

The custom query can be a stored procedure that accepts the poll quantity as input and returns an output parameter of type result set. Use the following syntax to specify a stored procedure:

```
call procedure_name (?, ?)
```

Where *procedure_name* is the name of the stored procedure to run. The first parameter represents the poll quantity, and the second parameter represents the result set.

The stored procedure can accept other input parameters as well, which you provide in the call statement itself, for example:

```
call procedure_name (25, ?, ?)
```

Stored function

The custom query can be a stored function that accepts the poll quantity as input and returns a result set. Use the following syntax to specify a stored function:

```
? = call function_name (?)
```

Where *function_name* is the name of the stored function that should be run. The first parameter represents the result set, and the second parameter represents the poll quantity.

The stored function can accept other input parameters, which you provide in the call statement itself, for example:

```
? = call function_name (?, 'abc')
```

Custom update and delete queries

Custom event processing also allows you to provide custom update and delete queries, which are run after each event is processed. You typically use an *update query* to ensure that a database record does not get picked up for processing during subsequent poll cycles. Use a *delete query* when database records need to be deleted after each event is processed. Both the update and delete queries are optional.

Update and delete queries are specified by the CustomUpdateQuery and CustomDeleteQuery activation specification properties, respectively. You can enter these queries as a standard SQL statement, a stored procedure, or a stored function. The syntax for the custom update or delete query is the same as that for the custom query. Update and delete queries take an input parameter for the event ID. The adapter provides the value of event ID at run time. The queries can also have additional input parameters, which you provide in the query syntax itself, in the same way as described for the custom event query.

Event store

The event store is a persistent cache where event records are saved until the polling adapter can process them. The adapter uses the event store to keep track of inbound requests as they make their way through the system. Each time a database record is created, updated, or deleted, the adapter updates the status of the event in the event store. The status of each event is continually updated by the adapter for recovery purposes until the events are delivered to a configured export on the server.

The adapter polls the event records from the event store at regular intervals. In each poll call, a number of events are processed by the adapter. Events are processed in ascending order of priority and ascending order of the event time stamp. In each poll cycle, the adapter picks up all new events. For each new event, the adapter retrieves the value set in the object key field for the event and then loads the business object that corresponds to the value specified in the object name field. After the object is loaded, the adapter sets the primary key values of the business object based on the value specified in the object key field. After setting the keys, adapter performs a retrieval of the object based on the keys. The business object or optional business graph is created from the retrieved information and is published to the export.

If you have associated a stored procedure with the RetrieveAll operation of the business object, you can define the mapping between the input parameters of the stored procedure and the business object attributes (generally, primary keys). If such a mapping is defined, then the adapter sets the input parameters for the stored procedure, invokes the stored procedure, and populates the object based on the results obtained from the stored procedure.

For stored procedures and functions, if you defined a mapping between the input parameters of the stored procedure or function and the business object attributes (generally using primary keys) using the RetrieveSP application-specific information, then the adapter sets the input parameters on the stored procedure, invokes the stored procedure, and populates the business object based on the results obtained from the stored procedure.

When the object_function column has the value `Delete`, which indicates that the object was deleted, the object is not retrieved from the database. The keys are set on the data object and the business object or optional business graph is created and delivered to the export.

If an event is successfully posted, the entry is deleted from the event store. For failed events, the entries remain in the event store and the event_status column is set to -1.

The table format and content of the event store are described Table 2.

Table 2. Definition of the event store database table

Column name	**Type**	**Description**
XID	String	The unique transaction ID (XID) value for assured once delivery.
event_id	Number	The unique event ID, which is a primary key for the table. This can have the same value as the object_key.
object_key	String	A string that contains keys of the record in the event store that is retrieved. This column cannot be `null`. Specify the value as one or more *key=value* pairs, separated by the semicolon character (;). Alternatively, you can specify only the values for the primary keys separated by the semicolon (;) character. In this case, the values must be specified in the same order as primary keys are defined in the business object.
object_name	String	The name of the business object or business graph. The business object (or the business object within the business graph) can be a hierarchical business object. Each business object or business graph refers to a table or view. This column cannot be `null`.

Table 2. Definition of the event store database table (continued)

Column name	Type	Description
object_function	String	The operation corresponding to the event (Delete, Create, Update, and so on). This column cannot be `null`.
event_priority	Number	Identifies the event priority. This value must be a positive integer. This column cannot be `null`.
event_time	Timestamp	Date and time when event was generated. The format is `mm/dd/yyyy hh:mm:ss`.
event_status	Number	The event status. This is initially set to the value for a new event and updated by the adapter as it processes the event. The status can have one of the following values: • 0: Identifies a new event. • 1: Identifies an event that has been delivered to an export. • -1: An error occurred while processing the event. This column cannot be `null`.
event_comment	String	Any comment associated with the event.

Business objects

A business object is a structure that consists of data, the action to be performed on the data, and additional instructions, if any, for processing the data. WebSphere Adapter for JDBC uses business objects to represent tables and views in the database as well as the results of database queries, stored procedures, and stored functions. Business objects can also create a hierarchy of objects from your database and group unrelated tables. Your component communicates with the adapter using business objects.

How the adapter uses business objects

An integrated application uses business objects to access a database. The adapter converts the business objects in outbound requests into JDBC API calls to access the database. For inbound events, the adapters converts the data in the events into business objects, which are returned to the application.

The adapter uses business objects to represent the following types of objects in a database:

- Tables and views
- Synonyms and nicknames
- Stored procedures and stored functions

Some business objects do not represent database objects. These include:

- Batch SQL business objects, which represent a series of user-defined insert, update, and delete statements
- Query business objects, which represent a user-defined SQL query to run against the database

- Wrapper business objects, which let you group unrelated table and view objects into a single business object

How data is represented in business objects

For table or view business objects

Each column in the table or view is represented by a simple attribute of the table or view business object. A *simple attribute* is an attribute that represents a single value, such as a String, Integer, or Date. Other attributes represent a child business object or an array of child business objects.

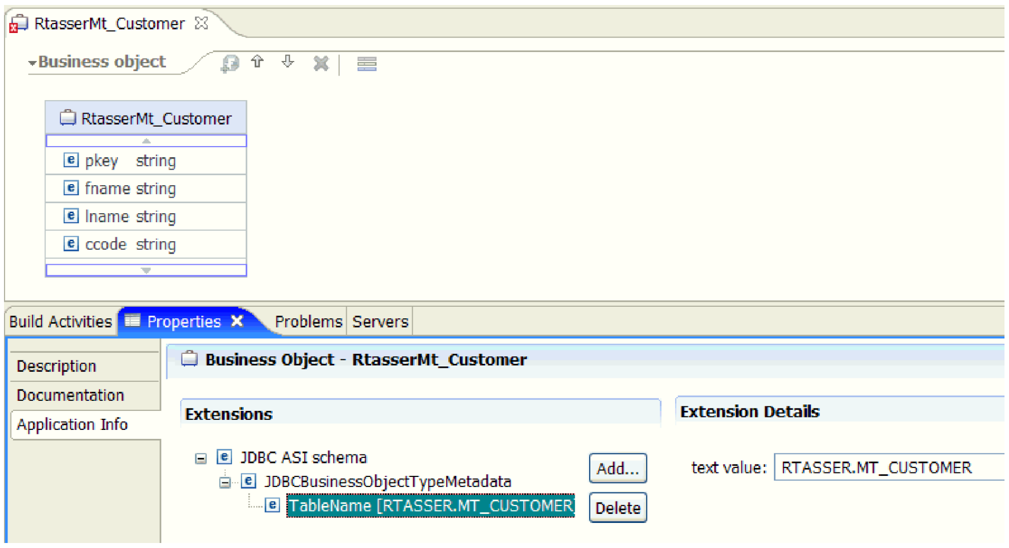
Simple attributes within the same business object cannot be stored in different database tables; however, the following situations are possible:

- The database table can 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 your application's processing of the business object should be included in your design.
- The business object can 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 are parameters for stored procedures or stored functions.
- The business object can represent a view that spans multiple database tables. The adapter can use such a business object when processing events triggered by changes to the database, such as Create, Update, and Delete operations. When processing business object requests, however, the adapter can use such a business object only for Retrieve and RetrieveAll requests.

A table business object always has a primary key, even if the corresponding
database table does not have a primary key. The adapter uses the column specified
2 in the primary key attribute when it retrieves table business objects. The adapter
2 supports tables that have composite, or multiple primary, keys. If a database table
has one or more primary keys, the wizard sets the primary key property for those
columns in the table business object. If the database table does not have a primary
key, the external service wizard prompts you for primary key information when
you configure that business object. Specify a column that contains unique data,
such as a sequence or identity column.

Table and view business objects support the Create, Update, Delete, Retrieve, RetrieveAll, and ApplyChanges outbound operations.

Figure 7 on page 24 shows a table business object in the business object editor. The business object has an attribute for each of the columns in the database table. Because the table has no child business objects, all of the attributes are simple attributes.


RtasserMt_Customer

Business object

RtasserMt_Customer

pkey	string
fname	string
lname	string
ccode	string

Build Activities | Properties | Problems | Servers

Description
Documentation
Application Info

Business Object - RtasserMt_Customer

Extensions

JDBC ASI schema
JDBCBusinessObjectTypeMetadata
TableName [RTASSER.MT_CUSTOMER]

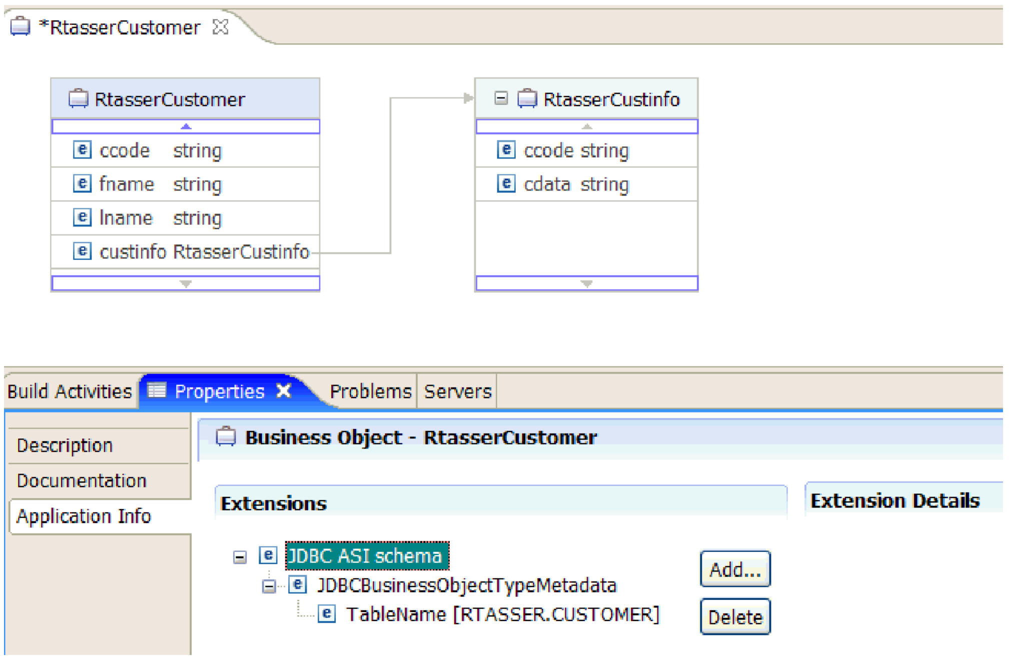
Add...
Delete

Extension Details

text value: RTASSER.MT_CUSTOMER

Figure 7. A table business object with no child.

Figure 8 shows a table business object that has one child table business object. The business object has simple attributes for each of the columns in the database table, plus a complex attribute pointing to a child business object.


*RtasserCustomer

RtasserCustomer

ccode	string
fname	string
lname	string
custinfo	RtasserCustinfo

RtasserCustinfo

ccode	string
cdata	string

Build Activities | Properties | Problems | Servers

Description
Documentation
Application Info

Business Object - RtasserCustomer

Extensions

JDBC ASI schema
JDBCBusinessObjectTypeMetadata
TableName [RTASSER.CUSTOMER]

Add...
Delete

Extension Details

Figure 8. A table business object with one child business object.

For stored procedure and stored function business objects

In a business object for a stored procedure or stored function, all of the input and output parameters for the stored procedure or stored function have corresponding attributes in the business object. If any of the input or output parameters is of a complex type, such as an array or structure, then the corresponding business object attribute is a child business object type with the child business object containing the attributes of the array or structure. If the stored procedure returns a result set, a child business object is created that contains the attributes of the returned result set.

The business object for stored procedures and stored functions supports the Execute outbound operation.

The sample file below shows the structure of stored procedure business objects. The business objects, ScottStrtValues and ScottStrtValuesStrt, are generated from a stored procedure that has one input type and two output types. One of the output parameters is of the Struct data type. The external service wizard generates a business object, ScottStrtValuesStrt, for the Struct type and adds it as a child object to the parent business object, ScottStrtValues. For the attribute of type Struct in the parent business object, the ChildBOType application-specific information is set to Struct to indicate it is of type Struct. The ChildBOTypeName application-specific information is set to the value of the user-defined Struct type in the database. The following examples shows the schema for the stored procedure.

Example of ScottStrtValues business object

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/
scottstrtvalues" xmlns:scottstrtvalues=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/scottstrtvalues" xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/
metadata" xmlns:scottstrtvaluesstrt=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/scottstrtvaluesstrt">
<import namespace=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/scottstrtvaluesstrt"
schemaLocation="ScottStrtvaluesStrt.xsd"/>
<import namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"
schemaLocation="JDBCASI.xsd"/>

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asiNSURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvalues">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
<jdbcasi:MaxNumOfRetRS>0</jdbcasi:MaxNumOfRetRS>
<jdbcasi:ResultSet>false</jdbcasi:ResultSet>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="pkey" type="int" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType>IP</jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="fname" type="string" minOccurs="0" maxOccurs="1">
<annotation>
```

```
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType>OP</jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="strt" type="scottstrtvaluesstrt:ScottStrtvaluesStrt"
minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType>OP</jdbcasi:SPParameterType>
<jdbcasi:ChildBOType>STRUCT</jdbcasi:ChildBOType>
<jdbcasi:ChildBOTypeName>STRUCT1</jdbcasi:ChildBOTypeName>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>
```

Example of ScottStrtValuesStrt business object

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/
scottstrtvaluesstrt" xmlns:scottstrtvaluesstrt=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/
scottstrtvaluesstrt" xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<import namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"
schemaLocation="JDBCASI.xsd"/>

<annotation>
<appinfo source="commonj.connector.asi">
<asi:annotationSet xmlns:asi="commonj.connector.asi" asiNSURI=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata"/>
</appinfo>
</annotation>
<complexType name="ScottStrtvaluesStrt">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCBusinessObjectTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPName>SCOTT.STRTVALUES</jdbcasi:SPName>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="name" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType></jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="title" type="string" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType></jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="dept_num" type="int" minOccurs="0" maxOccurs="1">
<annotation>
<appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
<jdbcasi:SPParameterType></jdbcasi:SPParameterType>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>
```

For query business objects

A business object for a database query defines the SQL statement that performs the query and the parameters that the query requires. The query business object supports the RetrieveAll outbound operation.

As an example, assume a query business object to run the following SELECT statement:

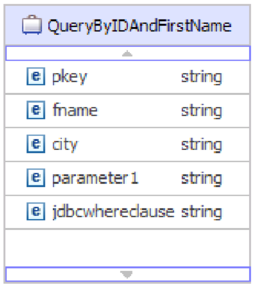
```
select C.pkey, C.fname, A.city from customer C, address A
    WHERE (C.pkey = A.custid) AND (C.fname like ?)
```

The question mark (?) indicates an input parameter for the query. A query can have multiple parameters, each indicated in the SELECT statement by a question mark. Table 3 shows the attributes of the sample query business object. The query business object has simple attributes for each column to be extracted, a simple attribute for each parameter, and a "placeholder object" for the WHERE clause of the query, which holds the WHERE clause after parameter substitution.

Table 3. Attributes of a query business object

Business object attribute	Description
pkey	Corresponds to database column PKEY in the Customer table
fname	Corresponds to database column FNAME in the Customer table
city	Corresponds to database column CITY in the Address table
parameter1	The parameter. There is one parameter for each ? (question mark) in the SELECT statement. In a SELECT statement with multiple parameters, subsequent parameters are named parameter2, parameter3, and so on.
jdbcwhereclause	A placeholder object for the WHERE clause

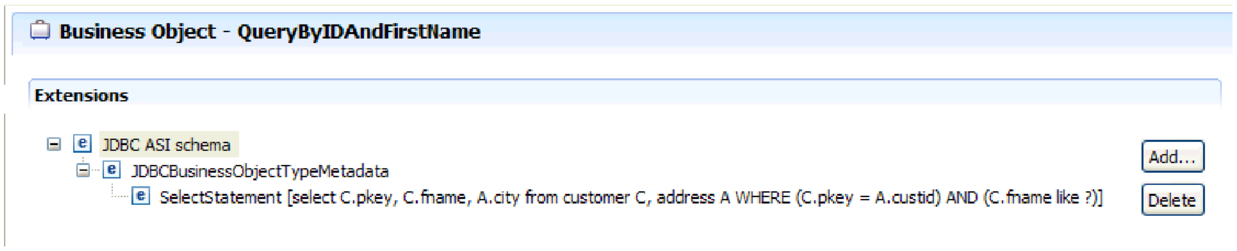
The following figure shows the business object for the sample query in the business object editor.



pkey	string
fname	string
city	string
parameter1	string
jdbcwhereclause	string

Figure 9. The attributes of a query business object

This figure shows the application-specific information for the query business object example. The SelectStatement application-specific information contains the SELECT statement.


Business Object - QueryByIDAndFirstName
Extensions
JDBC ASI schema
JDBCBusinessObjectTypeMetadata
Add...
Delete

SelectStatement [select C.pkey, C.fname, A.city from customer C, address A WHERE (C.pkey = A.custid) AND (C.fname like ?)]

Figure 10. The SELECT statement is saved in the business object application-specific information

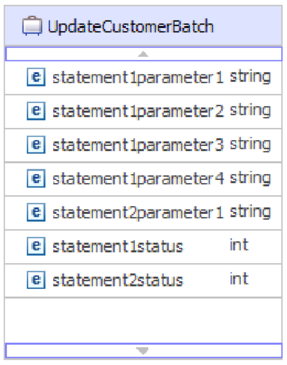
For batch SQL business objects

A batch SQL business object defines the INSERT, UPDATE, and DELETE SQL statements that perform the database actions and the parameters that the statements require. The batch SQL business object supports the Execute outbound operation.

As an example, assume a batch SQL business object to run the following INSERT and DELETE statements:

```
Insert into customer (pkey,ccode,fname,lname) values(?,?,?,?);
Delete From Customer where pkey=?
```

Each question mark (?) indicates a parameter for the statement. Each statement in a batch SQL business object can have multiple parameters, each indicated in the statement by a question mark. A batch SQL business object can have multiple statements, each with its own set of parameters.Figure 11 on page 29 shows the format of the business object for the batch SQL business object with an INSERT and a DELETE statement, each of which has one or more parameters.

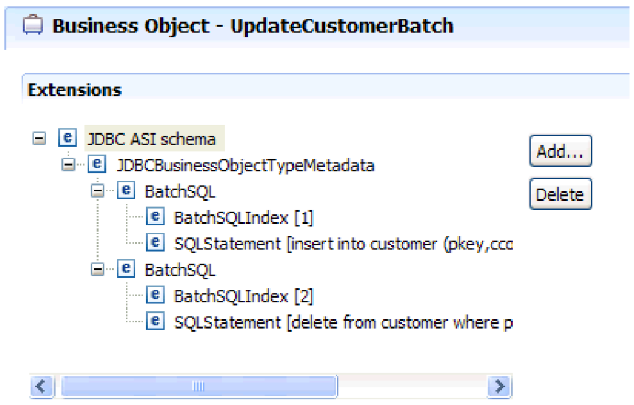


UpdateCustomerBatch

statement1parameter1	string
statement1parameter2	string
statement1parameter3	string
statement1parameter4	string
statement2parameter1	string
statement1status	int
statement2status	int

Figure 11. A batch SQL business object with two SQL statements

The business object has an attribute for each parameter in each statement, including statement1parameter1, statement2parameter1, and so on. It also has an attribute fir the status of each statement, such as statement1status, statement2status, and so on. The statements themselves are stored as application-specific information about the business object, as shown in Figure 12.



Business Object - UpdateCustomerBatch

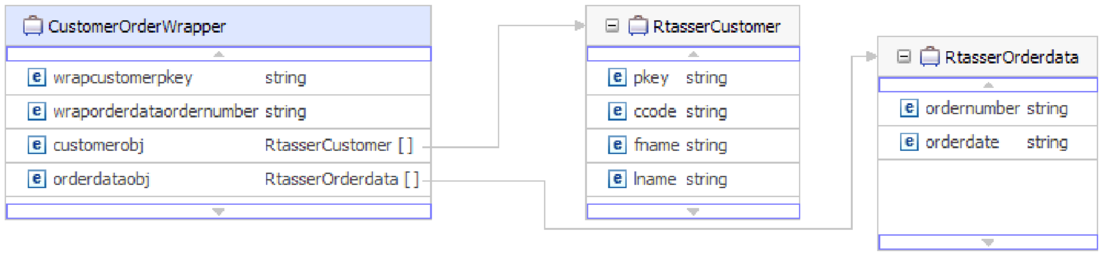
Extensions

```
JDBC ASI schema
  JDBCBusinessObjectTypeMetadata
    BatchSQL
      BatchSQLIndex [1]
      SQLStatement [insert into customer (pkey,ccc
    BatchSQL
      BatchSQLIndex [2]
      SQLStatement [delete from customer where p
```

Figure 12. The application-specific information of a batch SQL business object

For wrapper business objects

A wrapper business objects enable you to manipulate unrelated table and view business objects in a single operation. The wrapper business object supports the Create, Delete, Retrieve, and Update outbound operations.



CustomerOrderWrapper

wrapcustomerpkey	string
wraporderdataordernumber	string
customerobj	RtasserCustomer [ ]
orderdataobj	RtasserOrderdata [ ]

RtasserCustomer

pkey	string
ccode	string
fname	string
lname	string

RtasserOrderdata

ordernumber	string
orderdate	string

Figure 13. A wrapper business object that contains two table business objects

The wrapper business object contains a simple attribute for the primary key of each child business object. The name of the field is the string “wrap”, followed by the database table name and the column name of the primary key of the table. The wrapper business object also contains a complex attribute for each table business object. The name of the attribute is the table name with the string “obj” appended. The type of the complex attribute is the name of the corresponding table business object.

Business graphs

You can optionally choose, during adapter configuration, to generate a business graph. In version 6.0.2, each top-level business object is contained in a business graph, which includes a verb that an application can use in version 6.0.2 to specify additional information about the operation to be performed. In version 6.1.0, business graphs are required only in these situations:
- If you need to use the outbound ApplyChanges operation
- When adding business objects to a module created with a version of WebSphere Integration Developer earlier than version 6.1.0

If business graphs exist, they are processed, but the verb is ignored for all operations except ApplyChanges.

How business objects are created

You create business objects by using the external service wizard, launched from WebSphere Integration Developer. The wizard connects to the database, discovers database objects, and displays them to you. You select the database objects for which you want to create business objects. For example, you specify which schemas you want to examine. In those schemas, you select tables, views, stored procedures and functions, and synonyms and nicknames. In addition, you can create additional business objects. For example, you can create a business object to represent the results of user-defined SELECT, INSERT, UPDATE, or DELETE statements that are run against the database. The wizard helps you build a hierarchy of business objects, using parent-child relationships and wrappers for unrelated business objects.

After you specify which business objects you want and define the hierarchy of those objects, the wizard then generates business objects to represent the objects that you selected. It also generates other artifacts needed by the adapter.

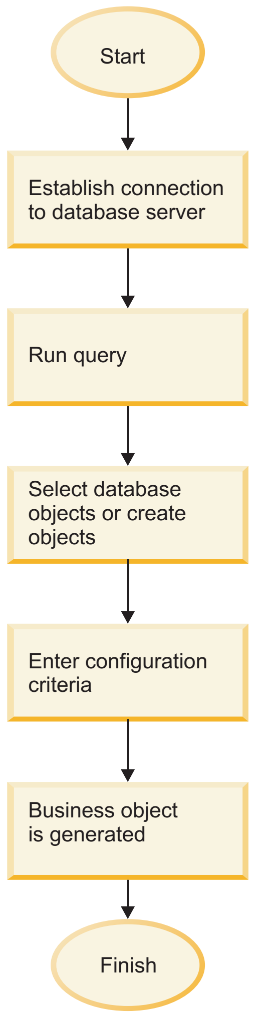

Start
Establish connection to database server
Run query
Select database objects or create objects
Enter configuration criteria
Business object is generated
Finish

Figure 14. How business objects are created

In some instances, the wizard cannot completely configure a parent-child relationship. For these relationships, you use the business objects editor, launched from WebSphere Integration Developer, to modify or complete the definition of a business object hierarchy that was created by the wizard. For more information, see the instructions for using the business object editor to modify business objects in the WebSphere Integration Developer information center at the following link: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/in.

Business object hierarchies

2 Define the relationships between database tables using parent-child relationships
2 and data ownership in hierarchical business objects. Unrelated tables can be
2 grouped with a wrapper business object.

Business objects can be either flat or hierarchical. In a flat business object, all attributes are simple and represent one row in the database table. Hierarchies can contain related or unrelated business objects. Related business objects have parent-child relationships, with or without ownership. Unrelated business objects use the wrapper business object.

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 parent business objects 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, which 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 child 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. The adapter supports single-cardinality relationships, and single-cardinality relationships and data without ownership.

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.

Use the following types of relationships between business objects to define a hierarchy that represents your database tables:
- Single-cardinality relationships
- Single-cardinality relationships and data without ownership
- Multiple-cardinality relationships
- Child business objects with multiple parents

In addition, unrelated business objects can be collected in a wrapper business object.

In each type of cardinality, the relationship between the parent and child business objects is described by the application-specific information of the key attributes in the business object storing the relationship.

Single-cardinality relationships in business objects:

In a single-cardinality relationship, 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. The adapter supports single-cardinality relationships, and single-cardinality relationships and data without ownership.

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.

Figure 15 on page 33 illustrates a typical single-cardinality relationship. In the example, FKey in the ParentBOName object is the simple attribute that contains the child's primary key, and Child(1), also in the ParentBOName object, is the attribute that represents the child business object.

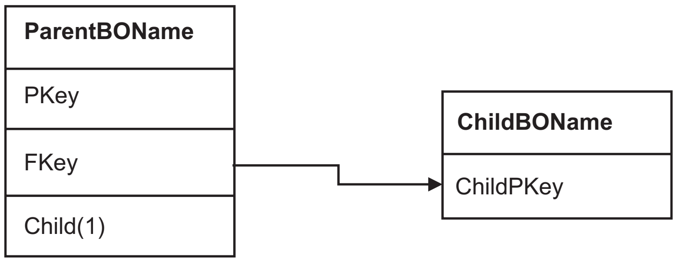


Figure 15. Typical single-cardinality relationship

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.

A parent business object can have a single-cardinality child with ownership and a single-cardinality child without ownership. Lookup tables are used for relationships without ownership. Ownership is indicated by the value of the Ownership application-specific information.

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 the Address database table contains a reference to the StateProvince lookup table. Because the lookup table is rarely updated and is maintained independently of the address data, creating or modifying address data does not affect the state and province data in the lookup table. However, to be able to retrieve the StateProvince business object along with the Address business object, StateProvince must be a single-cardinality child of Address and the relationship must be defined without data ownership.

If your database design includes lookup tables, your business object design will differ slightly from the database design. This is because the adapter retrieves data only for a table business object and its child table business objects. To use a lookup table, you need to create a single-cardinality parent-child relationship between the tables, without ownership. Although the StateProvince lookup table is not a child of the Address table in the database, the corresponding StateProvince business object is a single-cardinality child of the Address table business object because each address contains a single state or province. However, the Address business object does not "own" the StateProvince business object. Changes to an address do not result in a change to the list of states and provinces.

When the adapter receives a hierarchical business object with a Create, Delete, or Update request, the adapter does not create, delete, or update single-cardinality child business objects 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; it does not add or change values in the lookup table's business object.

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: When the relationship is without ownership, you can 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 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 a multiple-cardinality relationship, an attribute in the parent business object represents an array of child business objects. The attribute is of the same type as the child business object. The foreign key that describes the relationship is stored in the child, except when an application stores a single-child entity. Then the parent-child relationship is stored in the parent.

Typically, a business object that contains an array of child business objects has only one attribute that represents the relationship, and this attribute is normally the primary key. 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.

Figure 16 on page 35 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.

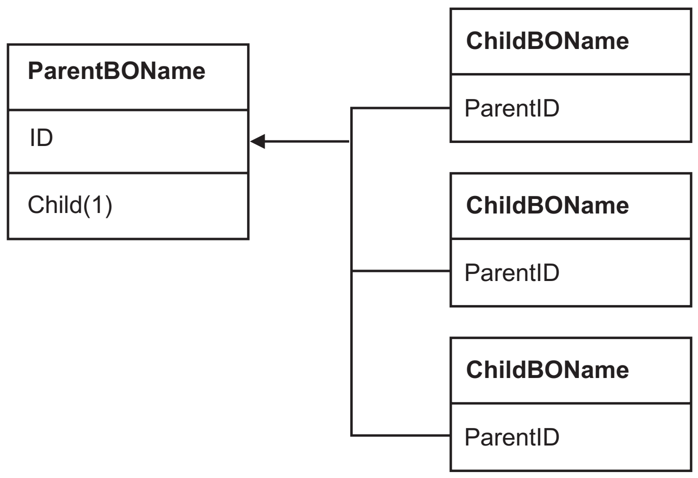


Figure 16. Multiple cardinality business object relationship with N>1

A multiple-cardinality relationship can 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. Figure 17 shows this type of relationship.

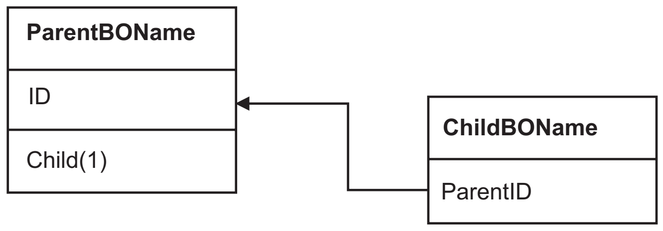


Figure 17. Multiple cardinality relationship with N=1

Database tables with multiple parent tables:

If a child table in the database has more than one parent table, you must manually configure additional parent business objects using the assembly editor. The external service wizard configures only one parent.

The business object schema

The business object schema is built out of database objects that you select when you run the external service wizard. Each database object translates into a top-level business object.

The schema defines business objects names and application-specific information. The business objects and their attributes and application-specific information are represented in the schema as follows:

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

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

Stored procedure overview

A stored procedure can be a business object that your module runs with the Execute operation, it can run in place of the standard SQL for an operation on any business object, or it can perform additional actions before or after performing an operation.

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 uses stored procedures in the following ways:

- By creating a stored procedure business object to run against your database
- By enhance a business object's operations by replacing the SQL statement provided for a business object's operation or by performing actions before or after the operation runs

Stored procedure business object overview

You can create a stored procedure business object that corresponds to a stored procedure or stored function in the database. You can then use the Execute operation to run the stored procedure against data in the database.

The external service wizard helps you build stored procedure business objects that run a stored procedure or stored function. The wizard examines the stored procedure or stored function in the database to create the business object. A stored procedure business object has an attribute for each parameter.

If a parameter attribute has a simple data type, there is an attribute for a sample value for the parameter. The wizard uses the sample values when you validate the stored procedure before saving it. The adapter uses the result from the stored procedure to validate the parameters, to obtain the maximum number of result sets returned, and to be able to use the metadata of these result sets to generate child business objects. The wizard generates the hierarchy for stored procedure business objects automatically if you validate the stored procedure business object.

If the stored procedure has input/output parameters or return value parameters that are complex data types such as Struct, Array, or ResultSet, you need to select the corresponding the data type for each such parameter in the wizard and provide the name of the corresponding user-defined type. For Struct or Array type parameters, you also provide the name of the corresponding user-defined type name, which is saved in the property SPComplexParameterTypeName.

For example, if you create one Struct object named Struct_TEMP in the database, and you set the type as one input parameter, then you need to set the value of this property to `Struct_TEMP`. The wizard uses this type name to determine the metadata to generate for the corresponding child business object. If the stored procedure returns ResultSet, you need to set the number of result sets returned from this stored procedure in the property MaxNumberOfResultSets. This value represents the maximum number of returned result sets that will be handled by the adapter runtime.

For stored procedure business objects, the wizard supports nested Struct and Array objects, and can support any number of layers of nested hierarchy. The wizard can generate corresponding child business objects for all of these nested Struct and Array objects.

Table 4. Complex data type properties for stored procedure business objects

Property name	**Type**	**Description**

SPComplexParameterType	String	Value can be one of:
		
		`Array`
		`ResultSet`
		`Struct`

SPComplexParameterTypeName	String	The name of the user-defined type. This property is required when the value of SPComplexParameterType is `Struct` or `Array`.
MaxNumberOfResultSets	Integer	The maximum number of returned result sets to be handled by the Adapter for JDBC runtime. The wizard creates this number of business objects.

Stored procedures used in place of or in addition to operations

You can specify that the adapter use a stored procedure in the database in place of, before, or after the SQL statements that the adapter uses to perform an operation. Each business object can have a different set of stored procedures used with each operation.

The adapter can use simple SQL statements for Create, Update, Delete, Retrieve, or RetrieveAll operations. The column names used in the SQL statements are derived from an attribute's application-specific information. The WHERE clause is constructed using key values specified in the business object. Each query spans one table only, unless posted to a view. However, you can replace or enhance the SQL statement provided by the adapter using stored procedures and stored functions.

The adapter can call a stored procedure or stored function in the following circumstances:

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

In a hierarchical business object, if you want the stored procedure to be performed for each business object in the hierarchy, you must separately associate a stored procedure with the top-level business object and each child business object or array of business objects. If you associate a stored procedure with the top level business object but do not associate it with each child business object, then the top-level business object is processed with the stored procedure, but the child business objects are processed using the standard SQL query.

Table 5 lists the application-specific information elements for a stored procedure and describes their purpose and use. A complete description of each element is provided in the sections that follow the table. A sample stored procedure definition is shown in "Stored procedure sample" on page 41.

Table 5. Application-specific information for stored procedures in table and view business objects

Descriptive name	**Element name**	**Purpose**
Stored procedure type	StoredProcedureType	The stored procedure type defines the type of stored procedure to be used, and this determines when the stored procedure is called, for example, before processing a business object.
Stored procedure name	StoredProcedureName	The name of the stored procedure that is associated with the appropriate StoredProcedureType.
Result set	ResultSet	This value specifies whether the stored procedure returns a result set. If the result set is returned, a multiple-cardinality child for the current business object is created using the values returned in the result set rows.
Parameters	Parameters	Each Parameters element describes one parameter for a stored procedure or stored function.
Return value	ReturnValue	A value that indicates it is a function call, not a procedure call, because the value is returned by the function.

Stored procedure type

The stored procedure type defines the type of stored procedure to be used, and this determines when the stored procedure is called, for example, before processing a business object.

Table 6. Stored procedure type element characteristics

Required	Yes
Default	None

Table 6. Stored procedure type element characteristics (continued)

Possible values	Can be one of: • Before*Operation*SP • After*Operation*SP • *Operation*SP *Operation* specifies one of the operation names: Create, Update, Delete, Retrieve, or RetrieveAll.
Bidirectional transformation supported	No
Property type	String
Usage notes	Stored procedure types associated with RetrieveAll apply to top-level business objects only. You can remove any selected application-specific information from the StoredProcedureType property. All of the corresponding operation application-specific information property groups are also removed.
Examples	• CreateSP: Performs the create operation • UpdateSP: Performs the update operation • BeforeCreateSP: Runs before creating a business object • AfterCreateSP: Runs after creating a business object • AfterDeleteSP: Runs after deleting a business object

Stored procedure name

The name of the stored procedure that is associated with the appropriate StoredProcedureType.

Table 7. Stored procedure name element characteristics

Required	Yes
Default	None
Bidirectional transformation supported	Yes
Property type	String

Result set

2 This value determines whether the stored procedure returns a result or not. If the
2 result set is returned, a multiple-cardinality child for the current business object is
2 created using the values returned in the result set rows.

Table 8. Result set element characteristics

Required	Yes
Default	None
Possible values	`True` `False`
Bidirectional transformation supported	No

Table 8. Result set element characteristics (continued)

Property type	Boolean
Usage notes	Oracle users: If your stored procedure returns a result set, use the business object editor after finishing the external service wizard to verify that this attribute is set to `true`. The Oracle JDBC driver does not always return this value correctly.

Parameters

There is one Parameters element for each parameter for a stored procedure or stored function. Each Parameters element defines the name and type of one parameter.

Table 9. Parameters element characteristics

Required	Yes
Default	None
Contents	Each Parameters element specifies the following information: • PropertyName: Specifies the name of the business object attribute to pass as the parameter. • Type: Specifies the type of the parameter, one of the following values: – `IP` for input only – `OP` for output only – `IO` for input and output – `RS` for result set
Bidirectional transformation supported	No
Property type	String
Usage notes	In the case of Oracle stored procedures, a result set can be returned only as an output parameter. In that case, one of the parameters must have the type `RS`, to indicate a result set.

Return value

A value that indicates it is a function call, not a procedure call, because the value is returned by the function.

Table 10. Return value element characteristics

Required	No
Default	None
Possible values	Can be `RS` or the name of a business object attribute or child business object.
Bidirectional transformation supported	No
Property type	String

Table 10. Return value element characteristics (continued)

Usage notes	If the returned value is RS, the returned value is a result set and is used to create the multiple-cardinality container corresponding to this business object. If the returned value is the name of an attribute, the value is assigned to that particular attribute in the business object. If the attribute is another child business object, the adapter returns an error. When you associate a stored procedure with a business object that is generated from a table or view, and if the stored procedure is a function, a value will be returned from this stored procedure. One ReturnValue application-specific information value is added to the operation application-specific information. The existence of this application-specific information implies that it is a function call and not a procedure call, because a value is being retuned by the function. If the value of this application-specific information is a business object attribute name, the returned value is assigned to that particular attribute in the business object. If the value of this application-specific information is another child business object, the adapter runtime returns an error. In summary, if the returned value is of a simple data type, the wizard enables you to bind one business object attribute to it, and the value of this application-specific information is set to the name of that business object attribute. But if the returned value is a result set, the wizard sets the value of this application-specific information to RS. **Note:** For an Oracle database, a result set should be returned as an output parameter, not as a returned value. The type of the output parameter is set to RS to indicate that this parameter is used to return a result set. **Important:** In a hierarchical business object, if you want the stored procedure to be performed for each business object in the hierarchy, you must separately associate a stored procedure with the top-level business object and each child business object or array of business objects. If you associate a stored procedure with the top level business object but do not associate it with each child business object, then the top-level business object is processed with the stored procedure, but the child business objects are processed using the standard SQL query.

Stored procedure sample

2 The following sample show the XML definition of the customer business object in
2 the RtCustomer.xsd file, showing the definition of the stored procedures for
2 RetrieveSP and AfterRetrieveSP for the Retrieve operation. The adapter runs the
2 RT.RETR_CUST stored procedure in place of the standard SQL to retrieve a table
2 business object. After the business object is retrieved, the adapter runs the
2 RT.CUSTINFO stored procedure.

```
2   <jdbcasi:JDBCBusinessObjectTypeMetadata
2      xmlns:jdbcasi="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
2      <jdbcasi:TableName>RTASSER.CUSTOMER</jdbcasi:TableName
2        <jdbcasi:Operation> <jdbc asi:Name>Retrieve</jdbcasi:Name>
2          <jdbcasi:StoredProcedures>
2            <jdbcasi:StoredProcedureType>AfterRetrieveSP</jdbcasi:StoredProcedureType>
2            <jdbcasi:StoredProcedureName>RT.CUSTINFO</jdbcasi:StoredProcedureName>
2              <jdbcasi:Parameters>
2                <jdbcasi:Type>IP</jdbcasi:Type>
2                <jdbcasi:PropertyName>pkey</jdbcasi:PropertyName>
2              </jdbcasi:Parameters>
2              <jdbcasi:Parameters>
2                <jdbcasi:Type>OP</jdbcasi:Type>
```

```
             <jdbcasi:PropertyName>fname</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
           <jdbcasi:Parameters>
             <jdbcasi:Type>OP</jdbcasi:Type>
             <jdbcasi:PropertyName>lname</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
           <jdbcasi:Parameters>
             <jdbcasi:Type>OP</jdbcasi:Type>
             <jdbcasi:PropertyName>ccode</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
         </jdbcasi:StoredProcedures>
         <jdbcasi:StoredProcedures>
         <jdbcasi:StoredProcedureType>RetrieveSP</jdbcasi:StoredProcedureType>
         <jdbcasi:StoredProcedureName>RT.RETR_CUST</jdbcasi:StoredProcedureName>
           <jdbcasi:Parameters>
             <jdbcasi:Type>IP</jdbcasi:Type>
             <jdbcasi:PropertyName>ccode</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
           <jdbcasi:Parameters>
             <jdbcasi:Type>OP</jdbcasi:Type>
             <jdbcasi:PropertyName>fname</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
           <jdbcasi:Parameters>
             <jdbcasi:Type>OP</jdbcasi:Type>
             <jdbcasi:PropertyName>lname</jdbcasi:PropertyName>
           </jdbcasi:Parameters>
       </jdbcasi:StoredProcedures>
   </jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
```

Stored functions overview

Some databases support stored functions in addition to stored procedures. Stored functions are similar to stored procedures except that they always return a value. The adapters supports them in a similar manner.

For Oracle databases, the adapter supports stored functions that a user creates with the CREATE FUNCTION statement. Although this type of function is sometimes called a *user-defined function* (UDF), that term more typically refers to a Java stored function or procedure, which the adapter does not support.

For DB2 databases, the adapter supports stored procedures that return a value. Do not confuse this functionality with user-defined functions in DB2, where the term refers to an extension or addition to the existing built-in functions of the SQL language. You can use DB2 user-defined functions in the SQL you provide for operations, queries, and batch SQL statements performed by the adapter, but a user-defined function created with the CREATE FUNCTION statement is not typically represented to the adapter as a business object.

A function call has the following syntax:

```
? = call FunctionName parameter_list
```

Contrast this to a stored procedure call, which has the following syntax:

```
call SPName parameter_list
```

You specify the attribute that contains the returned value by using the ReturnValue business object application-specific information. For more information on ReturnValue, see “Business object-level application-specific information” on page 173.

Query business object overview

Query business objects run a user-defined SELECT statement against the database and return the matching records in business objects.

The external service wizard helps you build query business objects that run user-defined SELECT statements against the database. You specify the SELECT statement, using `?` (the question mark) in place of any substitutable parameters in the SELECT statement. The wizard then provides an area where you specify the data type of each parameter and provide a sample value. The sample value must match data in the database because wizard uses the SELECT statement's results to create the query business object.

Before you save the configuration of the query in the wizard, you validate it. When you validate, the wizard runs the SELECT statement using the sample values. After obtaining the result set, the wizard analyzes the metadata to obtain the column name and column type of all columns. For each column of the returned result set, the wizard generates one corresponding attribute in the query business object. For each parameter in the WHERE clause, the wizard generates one **jdbcwhereclause** attribute in the query business object and sets this attribute's default value to be the WHERE clause. These attribute are used to generate one dynamic WHERE clause at run time to replace the default WHERE clause.

For example, assume you specify the following SELECT statement:

```
select * from customer where fname=? and age=?
```

This WHERE clause has two parameters, indicated by the question marks (?). Its first parameter has the data type **string**, to match the data type of the fname column. The second parameter has the data type **int**, matching the age column. If your database has a customer record where the fname column contains the string `Mike` and the age column contains the integer `27`, you can specify those values as sample values when configuring the query business object. The wizard configures the business object to correspond to the returned result set.

Batch SQL business object overview

A batch SQL business object runs one or more user-defined INSERT, UPDATE, and DELETE statements and returns the status of the statements.

The external service wizard helps you build batch SQL business objects that run a set of user-defined INSERT, UPDATE, and DELETE statements against the database. You specify the statements, using `?` (the question mark) in place of any substitutable parameters in the statements. The wizard then provides an area where you specify the data type of each parameter and provide a sample value. The wizard uses the sample values when you validate the business object before saving the configuration.

The batch SQL statement does not support dynamic WHERE clauses for UPDATE and DELETE statements. The adapter does accept complex statements, such as JOIN or SUBSELECT, although the adapter does not parse these statements.

If the batch SQL business object contains a single INSERT statement, it can retrieve an automatically generated ID or identity value for the inserted rows.

Each SQL statement in the business object returns a status value, which is placed in an attribute named Statement*N*Status. For example, the status of the first SQL statement in a business object is in Statement1Status; the second statement's status is in Statement2Status, and so on.

If a single INSERT statement fails, the adapter throws the exception. If one of the statements in a batch UPDATE fails to run properly, the JDBC driver throws the java.sql.BatchUpdateException. When this happens, the adapter rolls back the transaction so that none of the SQL statements will be committed in the database.

The external service wizard

Use the external service wizard in WebSphere Integration Developer to discover objects in a database; to generate batch SQL, query, and wrapper business objects; and to generate business objects from selected database objects. The wizard also generates the module and the service artifacts that enable the adapter to run as a Service Component Architecture (SCA) component.

Standards compliance

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

Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability. WebSphere Adapters are fully accessible and section 508-compliant. Accessibility features enable users with physical disabilities, such as restricted mobility or limited vision, to operate software products successfully. These features are built into the installation and administration features of WebSphere Adapters.

Administration

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

- Use screen-reader software and a digital speech synthesizer to hear what is displayed on the screen
- Use voice recognition software, such as IBM ViaVoice®, to enter data and to navigate the user interface
- Operate features by using the keyboard instead of the mouse

You can configure and use product features by utilizing standard text editors and scripted or command-line interfaces instead of the graphical interfaces that are provided.

When appropriate, the documentation for specific product features contains additional information about the accessibility of the features.

External service wizard

The external service wizard is the primary component used to create modules. This wizard, which is implemented as an Eclipse plug-in that is available through WebSphere Integration Developer, is fully accessible.

Keyboard navigation

This product uses standard Microsoft Windows® navigation keys.

IBM and accessibility

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

Internet Protocol Version 6 (IPv6)

WebSphere Process Server and WebSphere Enterprise Service Bus rely on WebSphere Application Server for Internet Protocol Version 6 (IPv6) compatibility. In addition, WebSphere Adapter for JDBC relies on the IPv6 support of the database server and the JDBC driver you use to connect to the server.

If the database server and JDBC driver support pure IPv6, then you can use pure IPv6 features with the adapter.

IBM WebSphere Application Server, version 6.1.0 and later support pure Internet Protocol Version 6.0 (IPv6).

For more information about this compatibility in WebSphere Application Server, see IPv6 support in the http://www.ibm.com/software/webservers/appserv/was/library/.

For more information about IPv6, see http://www.ipv6.org.

Chapter 2. Planning for adapter implementation

Before using WebSphere Adapter for JDBC, make sure you understand the experience you need and the server environment in which it runs. Learn the considerations for deploying the adapter in your server environment, and find out how to improve the performance and availability of the adapter by using a clustered server environment.

Before you begin

Before you begin to configure and deploy the module, you should possess a thorough understanding of business integration concepts, Java Database Connectivity (JDBC), the database products in your environment, and the features and capabilities of WebSphere Integration Developer and WebSphere Process Server or WebSphere Enterprise Service Bus.

To configure and deploy WebSphere Adapter for JDBC you should understand and have experience with the following concepts, tools, and tasks:

- The business requirements of the solution you are building.
- JDBC and the database products in your environment. This includes data access issues, transactional models, and connections across heterogeneous relational databases, queues, and Web services.
- Business integration concepts and models, including the Service Component Architecture (SCA) programming model.
- The capabilities and requirements of the server you plan to use for the integration solution. You should know how to configure and administer the host server and how to use the administrative console to set and modify property definitions, configure connection factories, and manage events.
- The tools and capabilities provided by WebSphere Integration Developer. You should know how to use these tools to create modules, wire and test components, and complete other integration tasks.

Security

The adapter uses the J2C authentication data entry, or authentication alias, feature of Java 2 security to provide secure user name and password authentication. For more information about security features, see the documentation for WebSphere Process Server or WebSphere Enterprise Service Bus.

User authentication

The adapter supports several methods for supplying the user name and password that are needed to connect to the database. Understand the features and limitations of each method to pick a method that provides the appropriate level of security and convenience for your application.

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

- When the external service wizard connects to the database to extract, or *discover*, information about the objects and services that you can access with the adapter.

- At run time on WebSphere Process Server or WebSphere Enterprise Service Bus, when the adapter connects to the database to process outbound requests and inbound events.

Authentication in the wizard

The external service wizard asks for connection information for both uses. You can use a different user name and password while running the wizard than you use when the application is deployed to the server. You can even connect to a different database, although the schema name must be the same in both databases. For example, while developing and integrating an application that uses Adapter for JDBC, you might not use the production database; using a test database with the same data format but fewer, simulated records lets you develop and integrate the application without impacting the performance of a production database and without encountering restrictions caused by the privacy requirements for customer data.

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

Authentication at run time

At run time, the adapter needs to provide the user name and password to connect to the database. To connect without user intervention, the adapter must access a saved copy of the user information. In a server environment, there are several methods for saving user information. The external service wizard lets you configure the adapter to get the user information using any of the following methods:
- Adapter properties
- Data source
- J2C authentication alias

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

Using a data source lets you use a connection already established for another application. For example, if multiple applications access the same database with the same user name and password, the applications can be deployed using the same data source. The user name and password can be known only to the first person who deploys an application to that data source or who defines a data source separately.

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

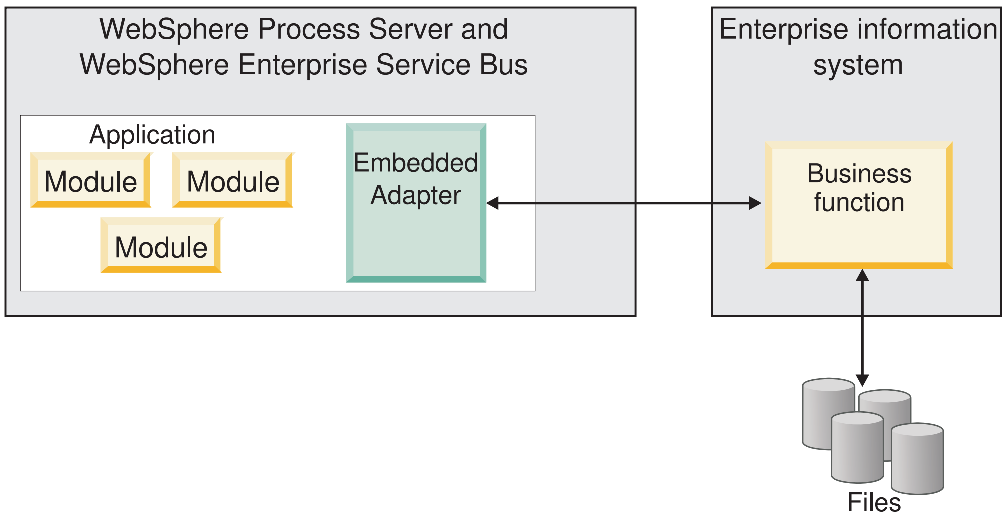
Deployment options

You can choose to embed the adapter to be part of the deployed application or you can choose to deploy the RAR file stand-alone.

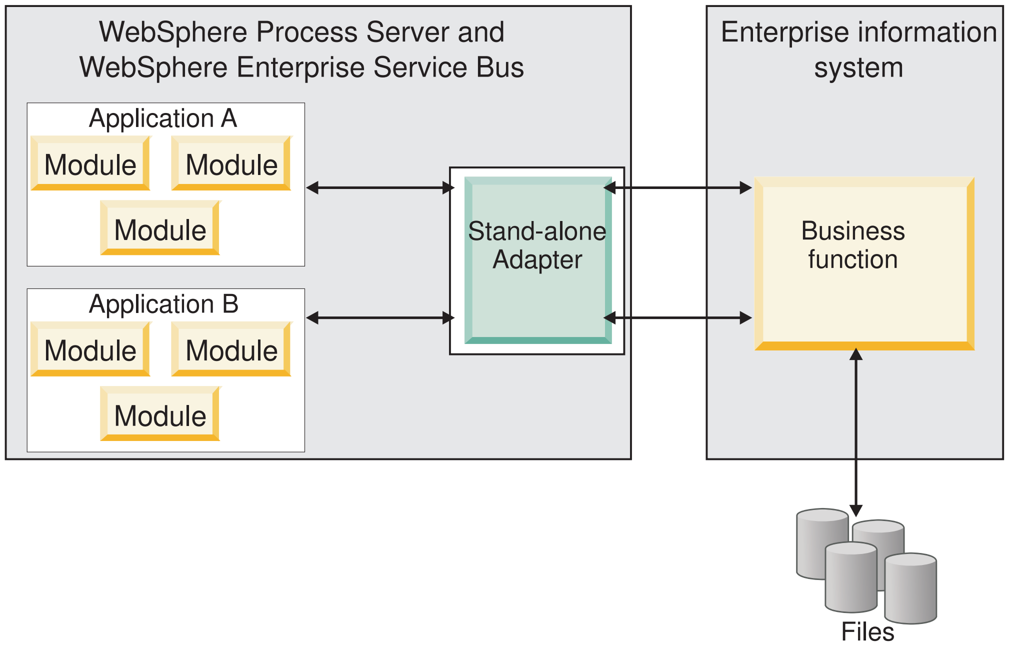
The deployment options are described below:

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

An embedded adapter is bundled within an enterprise archive (EAR) file and is available only to the application with which it is packaged and deployed.


WebSphere Process Server and WebSphere Enterprise Service Bus
Application
Module
Module
Module
Embedded Adapter
Enterprise information system
Business function
Files

A stand-alone adapter is represented by a stand-alone resource adapter archive (RAR) file, and when deployed, it is available to all deployed applications in the server instance.



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

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

If you plan on running multiple versions, and if you care more about potential disruption when you upgrade the adapter, you would be more likely to embed the adapter with the application. Embedding the adapter with the application allows you to associate an adapter version with an application version and administer it as a single module.

Considerations for embedding an adapter in the application

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

- An embedded adapter has class loader isolation.

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

Considerations for using a stand-alone adapter

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

- Stand-alone adapters have no class loader isolation.

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

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

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

WebSphere Adapters in clustered environments

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

WebSphere Process Server, WebSphere Application Server Network Deployment, and WebSphere Extended Deployment support clustered environments. Clusters are groups of servers that are managed together to balance workloads and to provide high availability and scalability. When you set up a server cluster, you create a Deployment Manager profile. The HAManager, a subcomponent of the Deployment Manager, notifies the JCA (Java EE Connector Architecture) container to activate the adapter instance. The JCA container provides a runtime environment for adapter instances. For information about creating clustered environments, see the following link: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun_wlm_cluster_v61.html.

Using WebSphere Extended Deployment, you can optionally enhance the performance of adapter instances in your clustered environment. WebSphere Extended Deployment extends the WebSphere Application Server Network Deployment capabilities by using a dynamic workload manager instead of a static workload manager, which is used by WebSphere Application Server Network Deployment. The dynamic workload manager can optimize the performance of adapter instances in the cluster by dynamically balancing the load of the requests. This means that application server instances can be automatically stopped and started based on the load variations, allowing machines with different capacities and configurations to evenly handle load variations. For information on the benefits of WebSphere Extended Deployment, see the following link: http://publib.boulder.ibm.com/infocenter/wxdinfo/v6r1/index.jsp.

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

High availability for inbound processes

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

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

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

High availability for outbound processes

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

If multiple outbound requests are attempting to process the same record, such as a Customer address, the workload management capability in WebSphere Application Server Network Deployment distributes the requests among the available adapter instances in the sequence they were received. As a result, these types of outbound requests in a clustered environment are processed in the same manner as those in a single server environment: one adapter instance processes only one outbound request at a time. For more information on workload management, see the following link: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/trun_wlm.html.

Support for prepared statement caching

WebSphere Adapter for JDBC supports prepared statement caching by the server, which can reduce the time required to perform an outbound or inbound operation or a batch of operations.

The adapter uses *prepared statements*, which are Java objects containing an SQL QUERY statement that is compiled once, but can be run multiple times. The server caches prepared statements to optimize their handling. If you want to use prepared statement caching for the adapter, define a data source using the administrative console, and enable caching on the data source. Then configure the adapter to use the data source, using one of these methods:

- Using the external service wizard when you initially configure the adapter to use the JNDI name of the data source
- Using the administrative console to set the DataSourceJNDIName property

Migrating to version 6.1.0

By migrating to version 6.1 of WebSphere Adapter for JDBC, you automatically upgrade from the previous version of the adapter. Additionally, you can migrate your applications that embed an earlier version of the adapter, so that the applications can utilize features and capabilities present in version 6.1.

Migration considerations

WebSphere Adapter for JDBC version 6.1.0 includes updates that might affect your existing modules.

Compatibility with earlier versions

WebSphere Adapter for JDBC version 6.1.0 is fully compatible with version 6.0.2 of the adapter and can work with custom business objects (XSD files), and data bindings created using WebSphere Adapter for JDBC version 6.0.2.

Because version 6.1 of the adapter is fully compatible with version 6.0.2, any of your applications that utilized version 6.0.2 of the adapter will run unchanged when you upgrade to version 6.1. However, if you want your applications to utilize features and functionality present in version 6.1 of the adapter, run the migration wizard.

The migration wizard replaces (upgrades) version 6.0.2 of the adapter with version 6.1 *and enables version 6.1 features and functionality for use with your applications*.

Note: The migration wizard does not create new or modify existing mitigating code, such as mappers and mediators to work with version 6.1 of the adapters. If any of your applications embed a 6.0.2.x or earlier version of an adapter and you are upgrading to version 6.1.0, and you want your applications to take advantage of the features and functions in 6.1, you might need to make changes to those applications.

If artifacts are inconsistent with regard to *versioning* within a single module, this module in its entirety will be marked as such, and will not be selectable for migration. Version inconsistencies are recorded in the workspace log, as this may be a symptom of project corruption.

The following scenarios are not supported:

- Running the external service wizard in WebSphere Integration Developer version 6.1.0 with WebSphere Adapter for JDBC version 6.0.2.
- Running the external service wizard in WebSphere Integration Developer version 6.0.2 with WebSphere Adapter for JDBC version 6.1.0.

Deciding whether to upgrade or to upgrade and migrate

The default processing of the migration wizard is to perform an upgrade of the adapter and to migrate the application artifacts so that the applications can utilize features and functions in version 6.1 of the adapter. When you choose to upgrade the connector by selecting a connector project, the wizard automatically selects the associated artifacts for migration.

If you decide that you want to upgrade the adapter from version 6.0.2 to version 6.1, but you do not want to migrate the adapter artifacts, you can do so by deselecting the adapter artifacts from the appropriate page of the migration wizard.

Running the migration wizard without any adapter artifacts selected will install and upgrade your adapter, but your artifacts are not migrated and your applications will not be able to take advantage of the features and capabilities that exist in version 6.1 of the adapter.

Run the migration wizard in a test environment first

Because adapter migration may require you to make changes to those applications that will utilize version 6.1 of WebSphere Adapter for JDBC, you should always perform the migration in a development environment first and test your applications before deploying the application to a production environment.

The migration wizard is fully integrated with the development environment.

Deprecated features

A deprecated feature is one that is supported but no longer recommended and that might become obsolete. Features from earlier versions of Adapter for JDBC that have been deprecated in version 6.1.0 include:

- Business graphs and verbs are now optional

  The business graph that contains each business object in version 6.0.2 is now optional. You need a business graph only for modules whose business objects were created in version 6.0.2 or for new version 6.1.0 modules that use the ApplyChanges outbound operation.
- Support for the UpdateWithDelete verb

Performing the migration

2 You can migrate a project or EAR file using the version 6.1.0, use the adapter
2 migration wizard. When the tool is finished, the migration is complete and you can
2 work in the project or deploy the module.

Before you begin

Review the information in *Migration considerations*.

About this task

To perform the migration in WebSphere Integration Developer, complete the following steps.

Note: After migration is complete, the module will no longer be compatible with previous versions of WebSphere Process Server, WebSphere Enterprise Service Bus, or WebSphere Integration Developer.

Note: The following steps describe how to run the adapter migration wizard from the connector project context menu while in the J2EE perspective in WebSphere Integration Developer.

Note: You can also migrate in one of the following ways:
- Right-click the project in the J2EE perspective and select **Migrate** → **Migrate project**.
- From the Problems view, right-click a migration-specific message and select **Quick Fix** to correct the problem.

Procedure

1. Import the PI (project interchange) file for an existing project or the EAR (enterprise archive) file for an deployed application into the workspace.
2. Change to the J2EE perspective.
3. Right-click the module and select **Migrate** → **Update Connector Project**.
4. Review the tasks and warnings presented on the welcome page, and then select **Next**.
5. On the Select Projects window, select **Next**.

   By default, the wizard migrates the connector project and any dependent projects. If your project has dependent projects and you do not want to migrate one or more of them at this time, clear their check boxes in the **Dependent adapter project** list. You can rerun the wizard to migrate the dependent project at a later time. Previously migrated projects, projects with a current version, and projects that contain errors are unavailable for migration and are not selected.
6. On the Adapter Migration window, optionally review the migration changes, but do not change any selections. Click **Finish**.
7. Check the Problems view for messages from the migration wizard, which start with the string CWPAD.
8. If you are migrating an EAR file, optionally create a new EAR file with the migrated adapter and artifacts, and deploy it to WebSphere Process Server or WebSphere Enterprise Service Bus. For more information about exporting and deploying an EAR file, see the topics devoted to it in this documentation.

Results

The project or EAR file is migrated to version 6.1.0. You do not need to run the external service wizard after exiting the adapter migration wizard.

Updating but not migrating a version 6.0.2 project

Before you can use a version 6.0.2 project, without migrating the complete project, with WebSphere Adapter for JDBC, version 6.1.0 in WebSphere Integration Developer, version 6.1.0, use the migration wizard to update the project, and then correct a problem.

About this task

Because the internal name of the adapter changed in version 6.1.0, artifacts in a version 6.0.2 project must be updated to use the new name before you can use the adapter wizard in WebSphere Integration Developer, version 6.1.0. Use the migration wizard to update a version 6.0.2 project. Then use the Quick Fix feature of WebSphere Integration Developer to change the adapter name in project artifacts.

Procedure

1. Import the project interchange (PI) file into the workspace.
2. In the J2EE perspective, right-click the project name and click **Migrate → Update Connector Project**. The adapter migration wizard opens.
3. On the welcome page, click **Next**.
4. On the Select Projects window, select none of the dependent artifact projects, and then click **Finish**.
5. In the Problems view, right-click the error message `CWPADL77A1: The IBM JDBC Adapter must be renamed...` and then click **Quick Fix**.
6. In the Quick Fix window, make sure the fix **Rename the referenced adapter** is selected, and then click **OK**.
7. If the error remains visible, click **Project → Clean**, select the project you just updated, and then click **OK**.

Results

The project can now be used with WebSphere Adapter for JDBC, version 6.1.0.

Chapter 3. Samples and tutorials

The WebSphere Integration Developer online samples/tutorials gallery includes samples and tutorials to help you use WebSphere Adapters.

You can access the online samples/tutorials gallery as follows:

- From the welcome page that opens when you start WebSphere Integration Developer. To see samples and tutorials for WebSphere Adapter for JDBC, click **Retrieve**. Then browse the displayed categories to make your selections.
- At this location on the Web: http://publib.boulder.ibm.com/bpcsamp/index.html.

© Copyright IBM Corp. 2006, 2007

Chapter 4. Configuring the module for deployment

To configure the adapter so that it can be deployed on WebSphere Process Server or WebSphere Enterprise Service Bus, use WebSphere Integration Developer to create a module, which is exported as an EAR file when you deploy the adapter. You then specify the business objects you want to discover and the system on which you want to discover them. After completing these steps, you will have successfully created an external service.

Roadmap for configuring the module

Before you can use WebSphere Adapter for JDBC in a runtime environment, you must configure the module. Understanding this task at a high level helps you perform the steps that are needed to accomplish the task.

You configure the module for the adapter to use by using WebSphere Integration Developer. The following figure illustrates the flow of the configuration task, and the steps that follow the figure describe this task at a high level only. See the topics following this roadmap for the details on how to perform each of these steps.

© Copyright IBM Corp. 2006, 2007

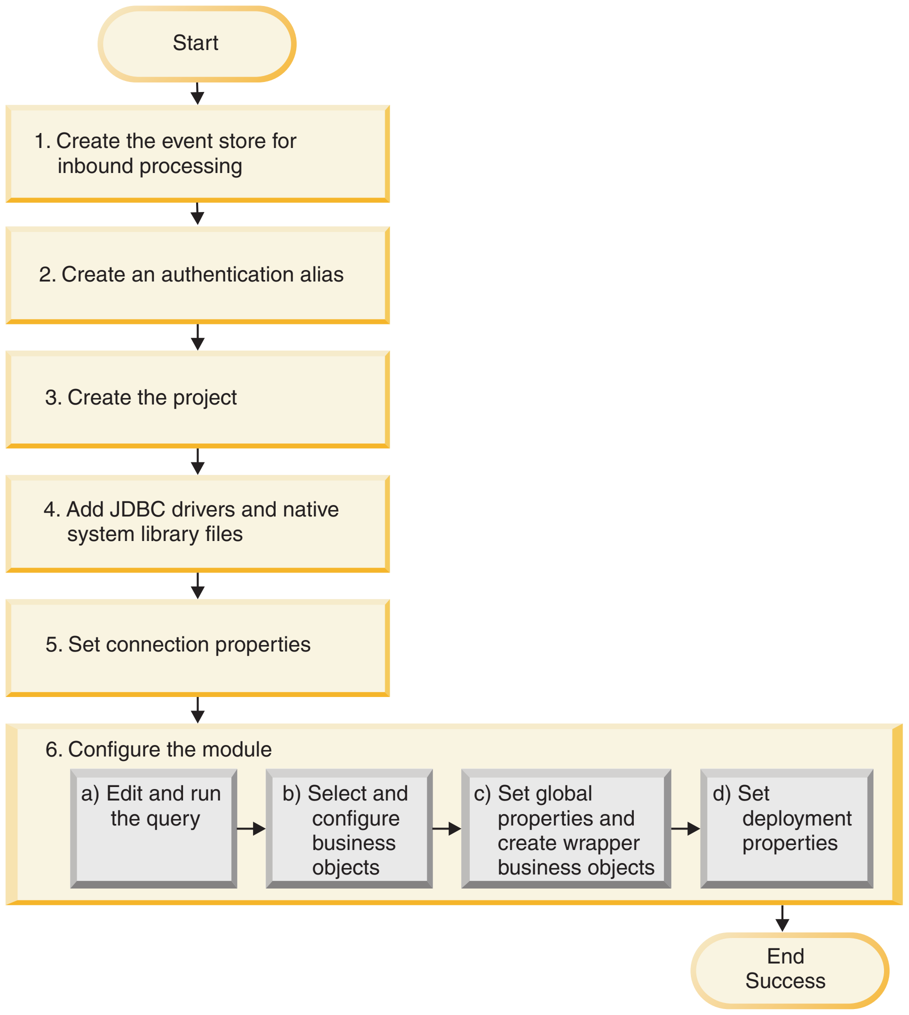

Start
1. Create the event store for inbound processing
2. Create an authentication alias
3. Create the project
4. Add JDBC drivers and native system library files
5. Set connection properties
6. Configure the module
a) Edit and run the query
b) Select and configure business objects
c) Set global properties and create wrapper business objects
d) Set deployment properties
End Success

Figure 18. Roadmap for configuring the module

Configuring the module for deployment

This task consists of the following high-level steps:

1. Create the event store for inbound processing.
2. Create an authentication alias to access the database with an encrypted password. This step is optional, depending on your policy for handling passwords and IDs. You perform this step using the administrative console on the server.
3. Create the project. First, start the external service wizard in WebSphere Integration Developer to begin the process of creating and deploying a module. The wizard creates a project that is used to organize the files associated with the module.
4. Add the JDBC drivers and native system library files required by WebSphere Adapter for JDBC to the project. These dependencies are also required when you export the module as an EAR file, and deploy the EAR file to the server.
5. Set connection properties that the external service wizard needs to connect to the database for discovery of objects and services.
6. Configure the module for inbound or outbound processing by using the external service wizard to find and select business objects and services from the database, and to generate business object definitions and related artifacts.

a. Edit and run the query that discovers database objects that you can access.
b. Select and configure business objects for inbound or outbound processing.
c. Set global properties for operations, and create wrapper business objects.
d. Set deployment properties that the adapter uses to connect to the database at run time. Then, generate the service by using the external service wizard to save the new module, which contains the business object or objects you configured, the import or export file, and the service interface.

Creating the event store

You need to create the event store in the database before the adapter can process inbound events. You can set triggers on user tables as needed to populate the event table.

About this task

Perform this task only if you need inbound processing of events. Create the event store in the database that contains the tables for which events are reported.

Procedure

1. Create the event store. Sample scripts are provided to create the event store for the IBM DB2, IBM DB2 for z/OS, Oracle, or Microsoft SQL Server database, as follows:
   - scripts_db2.sql
   - scripts_db2_zOS.sql
   - scripts_oracle.sql
   - scripts_mssql.sql

   These files are located in the *WID_installation_dir*/ResourceAdapters/JDBC_*version*/samples/scripts directory, where *WID_installation_dir* is the installation directory for WebSphere Integration Developer, and *version* identifies the version of the adapter, for example, `6.1.0.0_IF1`.
2. If necessary, set up triggers on user tables so that changes to the user tables can automatically generate events that are stored in the event store. The sample scripts contain examples of how to create triggers to populate your event store.

Results

The event store is available for event processing.

Creating an authentication alias

An authentication alias is a feature that encrypts the password used by the adapter to access the database. After an authentication alias has been created, you can use it when you configure the adapter (instead of directly typing the user ID and password). Adapter properties are not encrypted, and if you directly type password, it is stored as clear text that can be viewed by others. Using the authentication alias is the default choice in the external service wizard.

Before you begin

To create an authentication alias, you must have access to the administrative console of WebSphere Process Server or WebSphere Enterprise Service Bus. You must also know the user name and password to use to connect to the database.

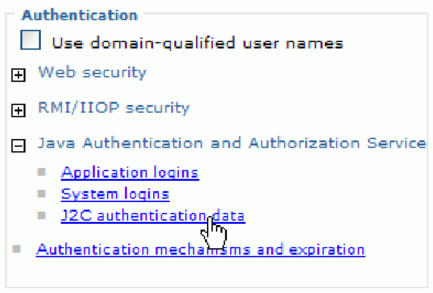
About this task

Using an authentication alias eliminates the need to store the password in clear text in an adapter configuration property, where it might be visible to others.

The following procedure shows you how to gain access to the administrative console through WebSphere Integration Developer. If you are using the administrative console directly (without going through WebSphere Integration Developer, log in to the administrative console and skip to step 2.

Procedure

1. Start WebSphere Integration Developer:
   a. Start WebSphere Integration Developer by clicking **Start** → **Programs** → **IBM Software Development Platform** → **IBM WebSphere Integration Developer 6.1** → **IBM WebSphere Integration Developer 6.1**.
   b. If you are prompted to specify a workspace, accept the default value. (The workspace is a directory where WebSphere Integration Developer stores your project.)
   c. When the WebSphere Integration Developer window is displayed, click **Go to the Business Integration perspective**.
2. Start the administrative console:
   a. In the Business Integration perspective, click the **Servers** tab.
   b. If the server does not show a status of **Started**, right-click the name of the server (for example, **WebSphere Process Server**) and click **Start**. Wait for the status of the server is `Started`.
   c. Right-click the name of the server and click **Run administrative console**.
   d. Log on to the administrative console. If your administrative console requires a user ID and password, type the ID and password and click **Log in**. If the user ID and password are not required, click **Log in**.
3. In the administrative console, click **Security** → **Secure administration, applications, and infrastructure**.
4. Under **Authentication**, click **Java Authentication and Authorization Service** → **J2C Authentication data**.


Authentication
Use domain-qualified user names
Web security
RMI/IIOP security
Java Authentication and Authorization Service
Application logins
System logins
J2C authentication data
Authentication mechanisms and expiration

5. Create an authentication alias
   a. In the list of J2C authentication aliases that is displayed, click **New**.
   b. In the **Configuration** tab, type the name of the authentication alias in the **Alias** field.
   c. Type the user ID and password that are required to establish a connection to the database.
   d. Optionally type a description of the alias.
   e. Click **OK**.

The newly created alias is displayed.

Note the full name of the alias, which includes the node name. This full name is the one you use in subsequent configuration windows.

f. Click **Save**.

Results

You have created an authentication alias, which you specify when you configure the adapter properties later in the wizard.

Creating the project

To begin the process of creating and deploying a module, you start the external service wizard in WebSphere Integration Developer. The wizard creates a connector project, which is used to organize the files associated with the module.

Before you begin

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

About this task

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

Procedure

1. If WebSphere Integration Developer is not currently running, start it now.
   a. Click **Start** → **Programs** → **IBM Software Development Platform** → **IBM WebSphere Integration Developer 6.1** → **IBM WebSphere Integration Developer 6.1**.
   b. If you are prompted to specify a workspace, accept the default value.

      The workspace is a directory where WebSphere Integration Developer stores your project.
   c. When the WebSphere Integration Developer window is displayed, click **Go to Business Integration Perspective**.
2. Start the external service wizard by clicking **File** → **New** → **External Service**.
3. In the New External Service window, select **Adapters** and click **Next**.
4. In the Select an Adapter window, select **IBM WebSphere Adapter for JDBC (IBM : *version*)**, where *version* is the version of the adapter you want to use, for example, 6.1.
5. Click **Next**.
6. In the Adapter Import window, accept the default project name in **Connector project** or type a different name.
7. In **Target runtime**, select the type of server where you will deploy the module. The wizard creates the artifacts that are appropriate to that server.
8. Click **Next**. The Required Files and Libraries window is displayed.

Results

A new connector project is created, which contains the adapter RAR file. The project is listed in the Business Integration perspective.

What to do next

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

Adding external software dependencies

The adapter needs a copy of certain files from the database to be able to communicate with it. Use the external service wizard to specify the location of the JAR files that contains the JDBC driver and any native system library files that are needed.

Before you begin

You should be running the external service wizard in WebSphere Integration Developer to perform this task.

About this task

In addition to performing this task when you configure the module, you might also need to deploy files on WebSphere Process Server or WebSphere Enterprise Service Bus.

Procedure

1. Obtain the JDBC driver-specific files and native libraries for your database software and operating system from your database administrator or from the database software Web site. The files that you need vary by database server. The following table lists the JDBC driver files needed for common database software.

Table 11. JDBC driver files for common database software

Database software	Driver	JDBC driver files	Native System Libraries
IBM DB2 Universal Database™ for Linux®, UNIX®, and Windows	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cu.jar	None
IBM DB2 for z/OS	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cisuz.jar	None
IBM DB2 for i5/OS®	IBM Toolbox for Java remote driver	jt400.jar db2jcc_license_cisuz.jar	None
	IBM DB2 Universal driver	db2jcc.jar	None
	IBM Toolkit for Java native driver*	db2_classes.jar	None
Oracle	Thin driver	ojdbc14.jar	None
Microsoft SQL Server 2005	Microsoft SQL Server 2005 for JDBC	sqljdbc.jar	None

Table 11. JDBC driver files for common database software (continued)

Database software	Driver	JDBC driver files	Native System Libraries
* You can use the IBM Toolkit for Java native driver to connect to the database at adapter run time, but you cannot use it to connect while running the wizard. You must use either the IBM Toolbox for Java remote driver or the IBM DB2 Universal driver during the discovery process. However, you can configure the module to use the native driver at run time. Do this on the Service Generation and Deployment Configuration window.			

2. In the Required Files and Libraries window, specify the location of the JDBC driver-specific files that are required by the project. The wizard will import the files into the project for use with the adapter.
   a. In **JDBC driver JAR files**, click **Add** and select the JDBC driver files.
   b. If you plan to use a JDBC type 2 driver, click **Add** in **System libraries** to add the native system libraries that are required to access the database server. If you will use only JDBC type 4 drivers, leave this field empty.
3. Click **Next**. The wizard displays the Processing Direction window.

Results

The connector project now contains the JDBC files needed by the module and an embedded adapter.

Continue working in the external service wizard. The next step is to provide the information that the wizard needs to connect to the database.

Setting connection properties for the external service wizard

Specify the connection properties that the external service wizard uses to connect to the database instance to discover database objects.

Before you begin

Before you can configure the connection properties, you must have started the external service wizard.

About this task

The external service wizard requires these properties to connect to the database for discovery and for creating the service description. For more information about the properties, see "Connection properties for the wizard" on page 179.

Procedure

1. In the Processing Direction window, select **Outbound** or **Inbound**, and then click **Next**.
2. In the Discovery Configuration window, specify the connection properties for the wizard to use to connect to the database.
   a. In the list of database software, select your product and version. The **Properties** area displays fields where you specify database-specific connection properties.

      Note: For IBM DB2 Version 9.1 for z/OS, select **V8 (New-Function Mode)**.
   b. In **JDBC driver type**, select the type of JDBC driver you want to use.

Note: For IBM DB2 for i5/OS, select `AS/400 Toolbox for Java` or `IBM DB2 Universal` to discover database objects. You can configure the module later to use the native driver for local access on the server at run time.

c. In **Database**, specify the database name. For Oracle databases, this is the System ID (SID).

d. In **Host name**, specify the host name or IP address of the database server. If you specify the IP address in IPv6 format, enclose the address in square brackets (`[]`).

e. In **Port number**, specify the port number for connecting to the database.

f. For DB2, Oracle, and Microsoft SQL Server databases, if you select a named driver in **JDBC driver type**, the wizard provides a default value for **JDBC driver classname** and builds the **Database URL** from other connection fields. If you select the driver `Other` for any database software and for certain other drivers, you must specify the driver class name and database URL (although part of the database URL might be filled in for you). See "Database URL" on page 181 and "JDBC driver classname" on page 182.

g. In **Additional JDBC driver connection properties**, specify additional properties to be set when connecting to the database. Specify one or more *name*:*value* pairs, separated by the semicolon character (;). For example:

```
loginTimeout:20;readOnly:true;securityMechanism:USER_ONLY_SECURITY
```

The connection information is used only for the discovery process. Later in the wizard, you can specify a different connection information to use at run time.

3. In **User name** and **Password**, type the user name and password to use to connect to the database from the wizard. This user name is used only during the discovery process and is not saved. Later in the wizard, you can specify a different user name and password or a different authentication method to use a run time.
4. In **Prefix for business object names**, type a string to be placed at the beginning of business object names.
5. To enable bidirectional support for the adapter at run time:
   a. Click **Advanced**.
   b. In **Bidi properties**, select **Bidi transformation**.
   c. Set the ordering schema, text direction, symmetric swapping, character shaping, and numeric shaping properties to control how bidirectional transformation is performed.
6. To change the location of the wizard's log files or the amount of information included in the logs, click **Change logging properties for wizard**, and then provide the following information:
   - In **Log file output location**, specify the location of the log file for the wizard.
   - In **Logging level**, specify the severity of errors that you want logged.

   This log information is for the wizard only; at run time, the adapter writes messages and trace information into the standard log and trace files for the server.
7. Click **Next**.

   If the wizard displays an error windows showing the com.ibm.adapter.framework.BaseException, it cannot connect to the database server. The message contains additional information about a possible cause for the problem. In addition, you can check the logs, which is located in the directory specified in **Log file output location**. Make sure that the connection information is correct.

Results

The external service wizard connects to the database and displays the Object Discovery and Selection window.

What to do next

Continue working in the wizard. The next step is to examine the database to locate the objects for which you want the wizard to create business objects.

Configuring the module for outbound processing

To configure a module to use the adapter for outbound processing, use the external service wizard in WebSphere Integration Developer to find and select business objects and services from the database, and to generate business object definitions and related artifacts.

Discovering database objects

After connecting to the database, run a query to search for database objects. Browse the tree of discovered objects to understand the structure of objects in the database and use filters to display only the database objects you want to see. Define how many business objects you want to create for user-defined database queries and for user-defined batch SQL statements.

Before you begin

You must understand the data requirement of the program that needs to access the database. For example, you need the following information about the database:

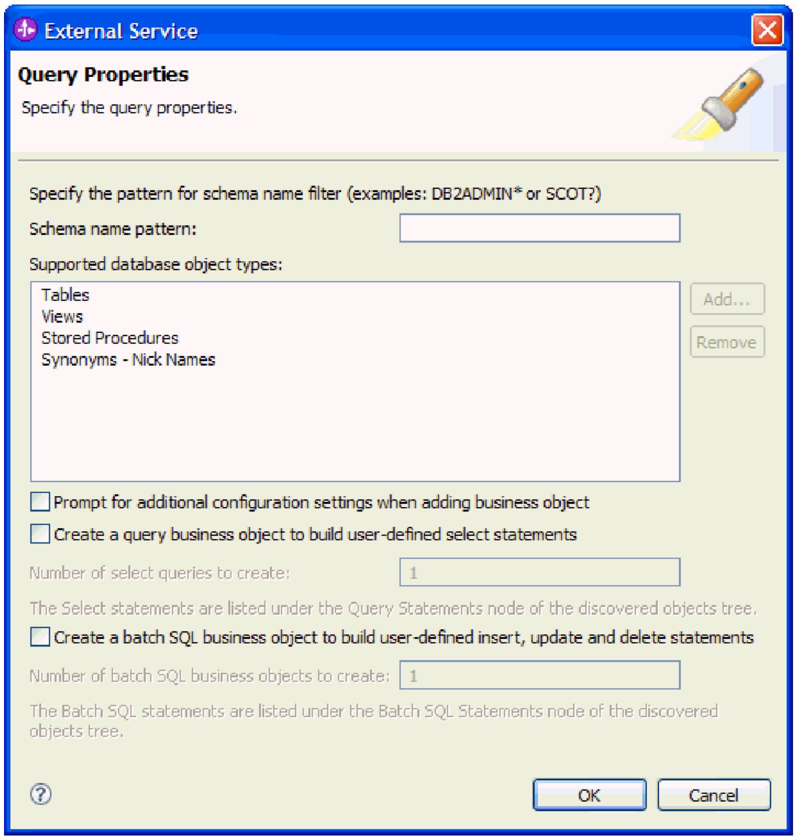
- Which schemas your module needs to access
- What type of database objects you need to access in those schemas
- Which tables, view, synonyms or nickname, or stored procedures or stored functions you need to access
- How many query and batch SQL business objects you need to create, including parameter values and sample database values for the parameters

About this task

This task starts in the Object Discovery and Selection window of the external service wizard.

Procedure

1. In the Object Discovery and Selection window, click **Edit Query**. The Query Properties window is displayed.


External Service

Query Properties

Specify the query properties.

Specify the pattern for schema name filter (examples: DB2ADMIN* or SCOT?)

Schema name pattern:

Supported database object types:

Tables
Views
Stored Procedures
Synonyms - Nick Names

- [ ] Prompt for additional configuration settings when adding business object
- [ ] Create a query business object to build user-defined select statements

Number of select queries to create: 1

The Select statements are listed under the Query Statements node of the discovered objects tree.

- [ ] Create a batch SQL business object to build user-defined insert, update and delete statements

Number of batch SQL business objects to create: 1

The Batch SQL statements are listed under the Batch SQL Statements node of the discovered objects tree.

Follow the instructions which follow to use the Query Properties window to perform the following tasks:

- Reduce the search time by searching a subset of database schemas
- Omit one or more types of database objects from the search
- Make the wizard prompt you for application-specific information that cannot be automatically determined based on information in the database
- Specify the number of query and batch SQL business objects you want to create

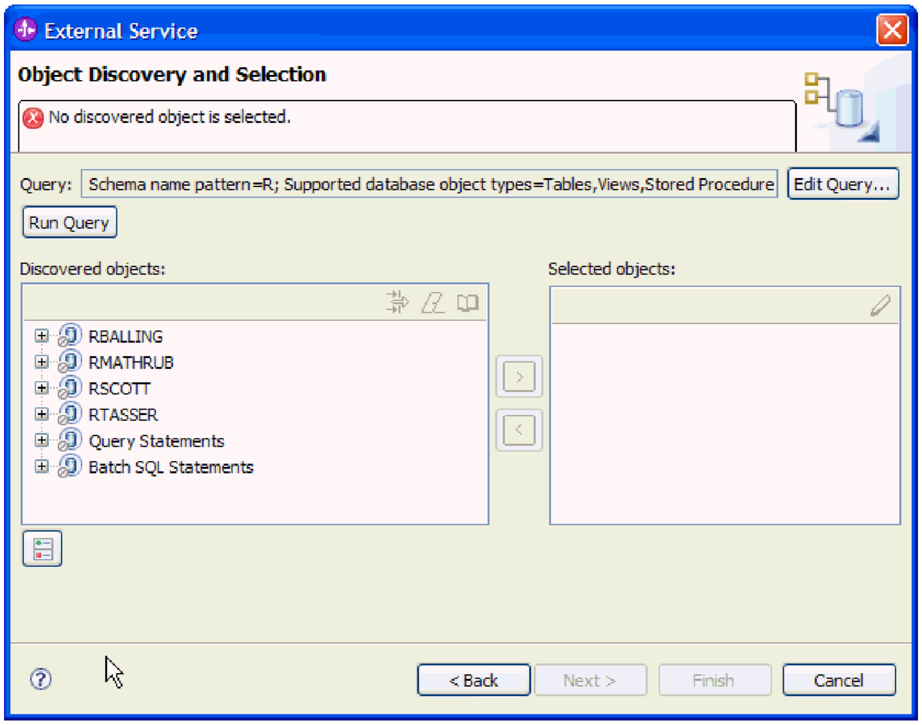
Note: In version 6.0.2 with fix pack 2, this window also allowed you to specify the number of wrapper business objects you want to create. In version 6.1.0, the wizard prompts for wrapper information at a later time.

2. To limit the number of database schemas that are retrieved, type the name of the schema or a name pattern in **Schema name pattern**. Use the question mark (?) character to match a single character and the asterisk (*) to match multiple characters. Only schemas that start with that string or match that pattern are displayed when you run the query. If you do not specify a schema name pattern, all schemas in the database are displayed. Using a filter can speed up the discovery process if your database contains many schemas.
3. To omit one or more types of objects from the search, select the types of objects that you want to omit (tables, views, stored procedures and stored functions, and synonyms or nicknames) in **Supported database object types**, and then click **Remove**. If you change your mind, click **Add** to add the object type back. If need to access only specific types of database objects, omitting the ones you do not need can speed up the discovery process.
4. Select **Prompt for additional configuration settings when adding business object**. Then, whenever you add a database object to the list of business

objects to create, the wizard automatically prompts you for all user-configurable application-specific information for the object. For example, if you select this option, the wizard guides you through the process of building a simple parent-child hierarchy of business objects. If you need hierarchy in which a table business object has two attributes which are referring to attributes in two different tables (that is, it has two parent business objects), you will need to complete the configuration in the assembly editor, a tool launched from WebSphere Integration Developer.

Important: If you do not select this option, the wizard prompts only for required information. You must complete the configuration of the business objects using the assembly editor.

5. To create business objects to run user-defined database queries, select **Create a query business object to build user-defined select statements** and then type the number of query business objects you want to create. You specify only the number of business objects at this time; the wizard prompts for the name and other details about the business objects at a later time.
6. To create business objects to run a sequence of SQL statements, select **Create a batch SQL business object to build user-defined insert, update and delete statements** and then type the number of batch SQL business objects you want to create. You specify only the number of business objects at this time; the wizard prompts for the name and other details about the business objects at a later time.
7. Click **OK** to save your changes to the database query.
8. In the Object Discovery and Selection window, click **Run Query** to use the query to discover database objects and to create templates for query and batch SQL business objects. The result of running a typical query are shown in the following figure.


External Service

Object Discovery and Selection

No discovered object is selected.

Query: Schema name pattern=R; Supported database object types=Tables,Views,Stored Procedure

The **Discovered objects** pane lists the database objects that were discovered.

9. In the **Discovered objects** list, click **+** (the plus sign) to expand a schema node and then expand **Tables**, **Views**, **Stored Procedures**, and **Synonyms - Nicknames** nodes beneath it, to see the database objects discovered by the wizard.
10. Click **+** (the plus sign) to expand the nodes for **Query Statements** and **Batch SQL Statements** to display the templates for query and batch SQL business objects.

Results

The wizard displays the database objects you can access using the adapter and business object templates for query and batch SQL business objects.

What to do next

Continue working in the external service wizard. The next step is to select the objects you want to use in your module, configure each business object, and create hierarchies of business objects.

Selecting and configuring business objects

Using the list of database objects discovered by the external service wizard, and the query and batch SQL object templates you specified, continue using the wizard to select the database objects that you need to access in your module. Then provide configuration information for your new business objects.

About this task

The Object Discovery and Selection window allows you to select and configure objects in any order, with the exception that you must select and configure a parent table before you can select and configure its child tables. Aside from this restriction, you have the flexibility to add objects individually or to add them several at a time. You can mix objects from the various nodes of the **Discovered objects** list. For example, you can select several table objects, a stored procedure object, and a Query statement, and add and configure them at the same time.

The high-level flow of selecting and configuring business objects is as follows:
1. You select one or more objects in the **Discovered objects** list of the Object Discovery and Selection window.
2. You click the > (Add) button.
3. The wizard opens the Configuration Properties window.
   - If you selected a single object, a single Configuration Properties window is displayed.

     You complete that window, specifying any user-configurable attributes and other information that the wizard cannot discover by examining the database, and click **OK** to save the configuration.
   - If you selected multiple objects, a Configuration Properties notebook is displayed, with one page for each object selected.

     You click on the name of each object in turn. The notebook displays the same information you would see had you selected this object individually.

     Important: Do not click **OK** on the notebook until you complete the configuration pages for all of the objects. The wizard will not close the notebook until you provide all of the required fields, but you can close the notebook before providing optional fields. If you do not configure the optional fields in the wizard, you must use the business object editor to configure them after exiting the wizard.

4. The wizard adds the configured object to the **Selected objects** list.

As long as you do not exit from the wizard, you can work iteratively to select and configure the business objects you need in your module. However, you cannot use the wizard to add objects to an existing module, so be careful to understand the requirements of the program that uses the business objects before you start the wizard.

Selecting and configuring tables, views, and synonyms or nicknames

Select and configure business objects for tables, views, and synonyms or nicknames for use in your module.

Before you begin

To select and configure business objects for tables, views, and synonyms or nicknames, you need to understand the structure of the data in the database and which database objects the module needs to access. This includes the following types of information:

- The structure of the tables, views, and synonyms or nicknames, including columns and column attributes such as data type
- The relationships between the tables, including the cardinality and ownership of parent-child relationships

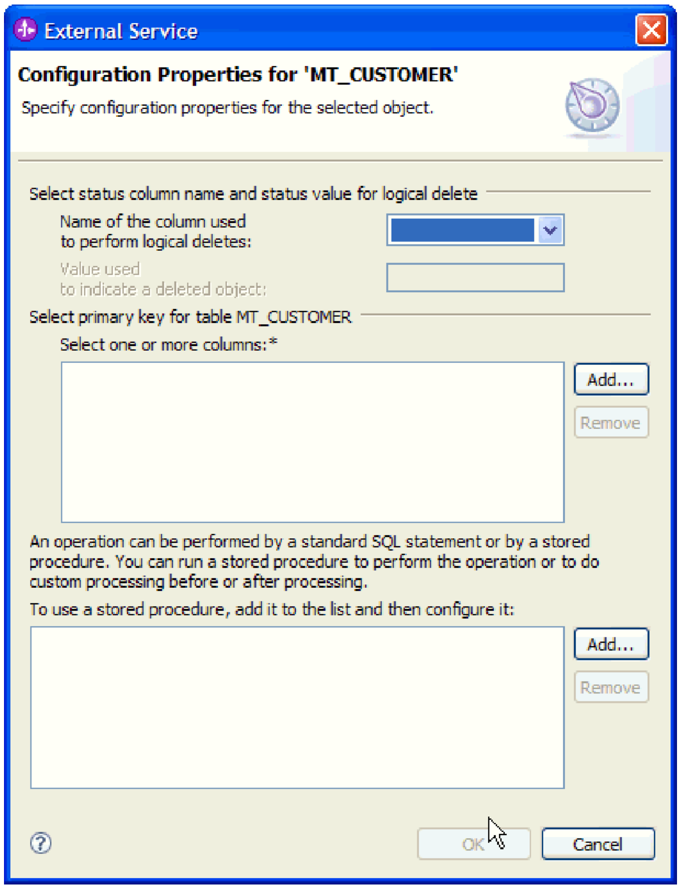
About this task

This task is performed using the external service wizard. You start in the Object Discovery and Selection window and then work in a Configuration Properties window that is specific to the business object you are configuring.

Procedure

1. In the **Discovered objects** list of the Object Discovery and Selection window, select one or more tables, views, or synonym, and click the > (Add) button to add the object to the **Selected objects** list.

   The following pair of figures shows a typical Configuration Properties window for a table, view, synonym, or nickname business object. The following figure shows a typical window for the first table or group of tables that you select.


External Service

Configuration Properties for 'MT_CUSTOMER'

Specify configuration properties for the selected object.

Select status column name and status value for logical delete

Name of the column used to perform logical deletes:

Value used to indicate a deleted object:

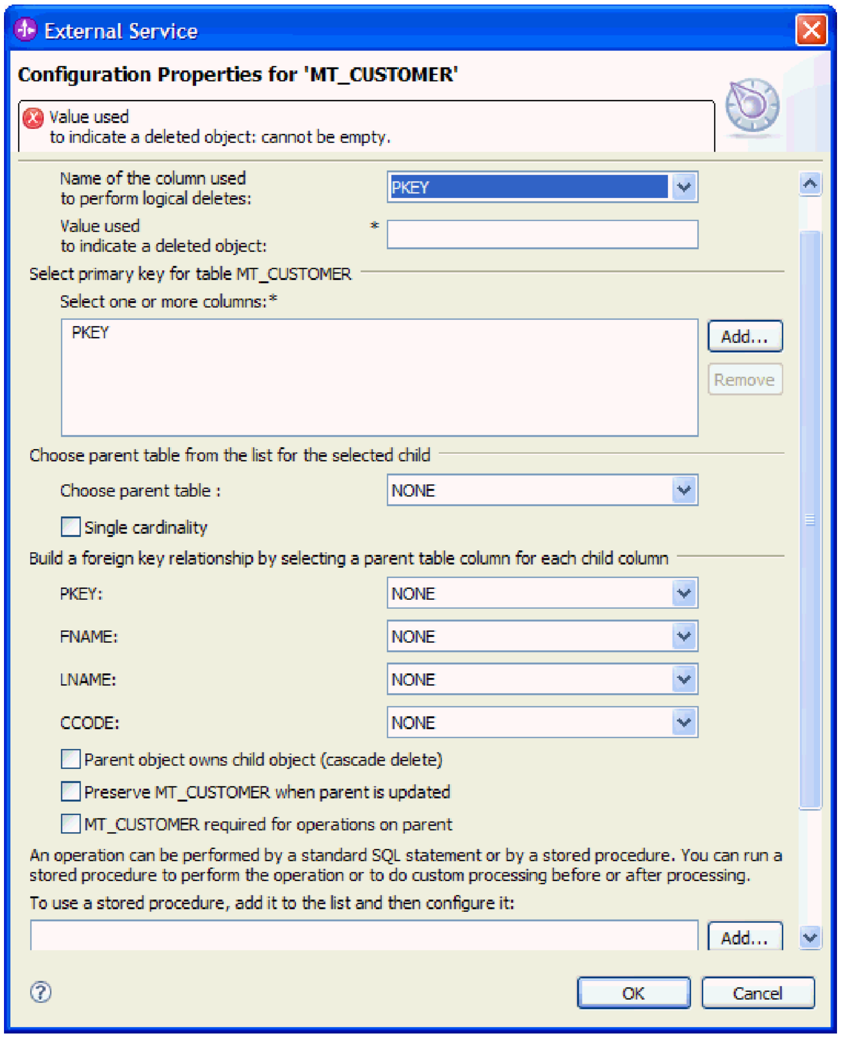
Select primary key for table MT_CUSTOMER

Select one or more columns:*

An operation can be performed by a standard SQL statement or by a stored procedure. You can run a stored procedure to perform the operation or to do custom processing before or after processing.

To use a stored procedure, add it to the list and then configure it:

The following figure shows a typical window for subsequent tables. After you select and configure at least one table, the Configuration Properties window for subsequent tables displays an area where you can optionally define a parent-child hierarchy between tables.


External Service

Configuration Properties for 'MT_CUSTOMER'

Value used
to indicate a deleted object: cannot be empty.

Name of the column used to perform logical deletes: PKEY

Value used to indicate a deleted object: *

Select primary key for table MT_CUSTOMER

Select one or more columns:*

PKEY

Choose parent table from the list for the selected child

Choose parent table : NONE

Single cardinality

Build a foreign key relationship by selecting a parent table column for each child column

PKEY:	NONE
FNAME:	NONE
LNAME:	NONE
CCODE:	NONE

Parent object owns child object (cascade delete)

Preserve MT_CUSTOMER when parent is updated

MT_CUSTOMER required for operations on parent

An operation can be performed by a standard SQL statement or by a stored procedure. You can run a stored procedure to perform the operation or to do custom processing before or after processing.

To use a stored procedure, add it to the list and then configure it:

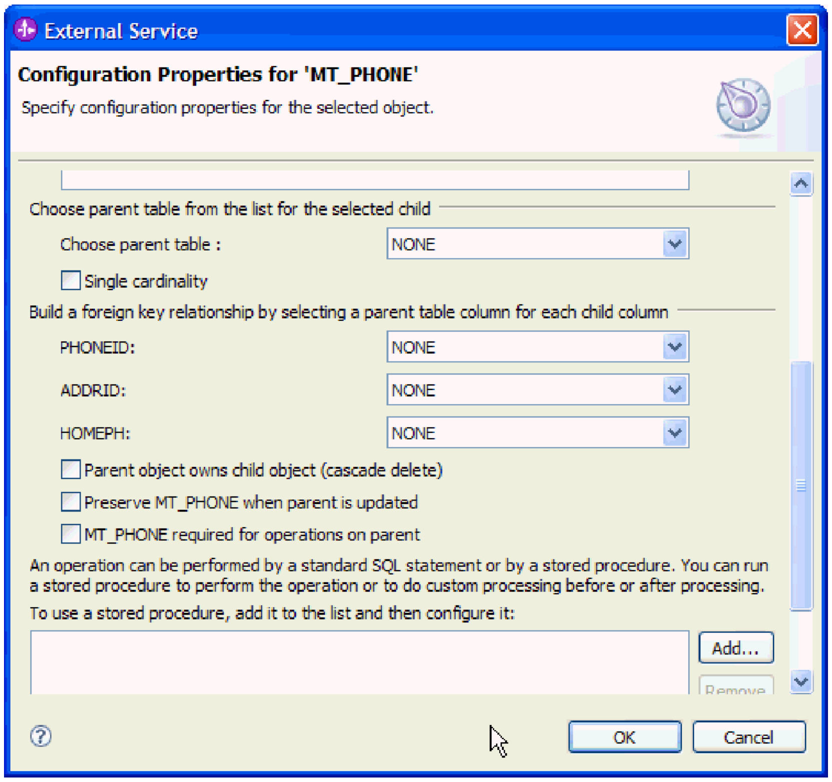
As you configure the object, choices that require advanced configuration might cause additional fields to be displayed in this window. This might cause the window to scroll. Be sure that you examine all fields on the window before clicking **OK**.

2. If the table has a column that is used to indicate logical deletes:
   a. Select the column name in **Name of the column used to perform logical deletes**.
   b. In **Value used to indicate a deleted object**, type the value that indicates that a row is logically deleted. You can get this value from your database administrator.
3. The **Select primary key for table** ***table_name*** area is displayed only when the database table does not have a column designated as the primary key. Each table business object must have a primary key, even if the associated database table does not have a key. If the primary key is defined in the database, this section of the window is not displayed.

   In **Select primary key for table** ***table_name***, click **Add**, select the column to be used as the primary key for the table business object, and then click **OK**. If the table has a composite key, you can select multiple columns.
4. Optionally, define a parent-child relationship between business objects.

Important: To build a parent-child hierarchy, configure the parent table first, and then return to the Object Discovery and Selection window to select and configure the child tables.

Configure the parent-child relationship using the area of the Configuration Properties window shown in the following figure. These fields are not displayed for the first table you configure.



External Service

Configuration Properties for 'MT_PHONE'

Specify configuration properties for the selected object.

Choose parent table from the list for the selected child

Choose parent table : NONE

☐ Single cardinality

Build a foreign key relationship by selecting a parent table column for each child column

PHONEID:	NONE
ADDRID:	NONE
HOMEPH:	NONE

☐ Parent object owns child object (cascade delete)

☐ Preserve MT_PHONE when parent is updated

☐ MT_PHONE required for operations on parent

An operation can be performed by a standard SQL statement or by a stored procedure. You can run a stored procedure to perform the operation or to do custom processing before or after processing.

To use a stored procedure, add it to the list and then configure it:

a. In **Choose parent table**, select the name of the table that is the parent of the table you are configuring. If you do not see the parent table in the list, the parent table has not yet been configured. Go back and configure the parent object before configuring the child objects.

b. Specify the cardinality of the relationship:

- If the table has a single-cardinality relationship with the parent table, select **Single-cardinality**. In a single cardinality relationship, a parent can have only one child business object of this type. A single-cardinality relationship can be used with ownership to represent a true child, or without ownership to represent lookup tables or other peer objects in a database.
- If the table is a multiple-cardinality relationship, do not select **Single cardinality**. In a multiple-cardinality relationship, a parent can have an array of child business objects of this type.

c. Build the foreign key relationship between the parent and child by specifying, for each child column, whether it is a foreign key in the parent table.

- If the child column is not a foreign key, select `NONE`.
- If a child column is a foreign key, select the column in the parent table that corresponds to the child column.

Note: The wizard can configure only a single parent table. If the child table has multiple parent tables, you must use the business object editor to configure the remaining parent tables after existing the wizard.

d. If the parent object owns the child object, then the child objects in the database are deleted when the parent is deleted. To indicate that this child is owned by its parent, select **Parent owns child object (cascade delete)**. Otherwise, clear this option to prevent child objects, such as lookup tables, from being deleted when their parent is deleted.

e. If you do not want child objects to be deleted as part of an Update operation, select **Preserve *child_table_name* when parent is updated**.

   When a parent table is updated, the adapter compares the child business objects present in the input with the child business objects returned from the database. By default, the adapter deletes any child objects returned from the database that are not present in the input business object.

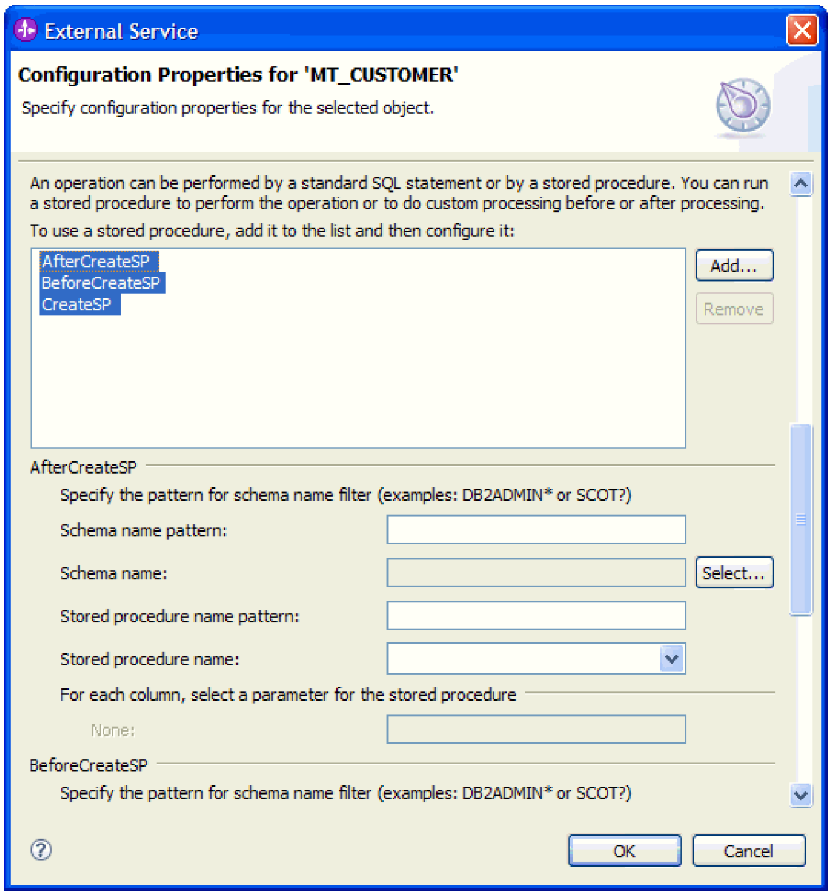
f. By default, you can perform operations on parent business objects without specifying the child business objects. If you want to requires that a parent business object must specify its child business objects when the parent is submitted for a change, select ***Child_table_name* required for operations on parent**.

5. An operation can be performed using either a standard SQL statement generated by the adapter or using stored procedures or stored functions from the database. If you want to use stored procedures or stored functions:

   a. Click **Add**.

   b. In the Add window, select the type of the stored procedure you want to run. For each operation, you can select a stored procedure that performs the operation, as well as stored procedures that run before or after the operation. For example, for the Create operation, you can specify any of CreateSP, BeforeCreateSP, and AfterCreateSP.

   c. Click **OK**. The Configuration Properties window now shows the stored procedure types you selected, and expands to display an area where you configure each one. It might be necessary to scroll down to see the new areas.


External Service

Configuration Properties for 'MT_CUSTOMER'

Specify configuration properties for the selected object.

An operation can be performed by a standard SQL statement or by a stored procedure. You can run a stored procedure to perform the operation or to do custom processing before or after processing.

To use a stored procedure, add it to the list and then configure it:

AfterCreateSP
BeforeCreateSP
CreateSP

AfterCreateSP

Specify the pattern for schema name filter (examples: DB2ADMIN* or SCOT?)

Schema name pattern:

Schema name:

Stored procedure name pattern:

Stored procedure name:

For each column, select a parameter for the stored procedure

None:

BeforeCreateSP

Specify the pattern for schema name filter (examples: DB2ADMIN* or SCOT?)

Important: In a hierarchical business object, if you want the stored procedure to be performed for each business object in the hierarchy, you must separately associate a stored procedure with the top-level business object and each child business object or array of business objects. If you associate a stored procedure with the top level business object but do not associate it with each child business object, then the top-level business object is processed with the stored procedure, but the child business objects are processed using the standard SQL query.

6. For each stored procedure type that you selected, specify the name of the stored procedure in the database and then configure the business object.
   a. Specify the name of the schema that contains the stored procedure. There are two ways to specify the name.
      - Filtering the database schemas to search for in the database.
        1) In **Schema name pattern**, type the name of the schema or a name pattern. Use the question mark (?) character to match a single character and the asterisk (*) to match multiple characters. The name is case-sensitive.
        2) In **Schema name**, select the name of the desired schema.
      - Selecting the schema from a list of all schemas discovered in the database.
        1) Leave the **Schema name pattern** field empty.
        2) Next to **Schema name**, click **Select** to open the Select window.
        3) In the Select window, select the name of the desired schema name. Optionally, type a case-sensitive pattern in **Value** to narrow down the

list. Use the question mark (?) character to match a single character and the asterisk (*) to match multiple characters.

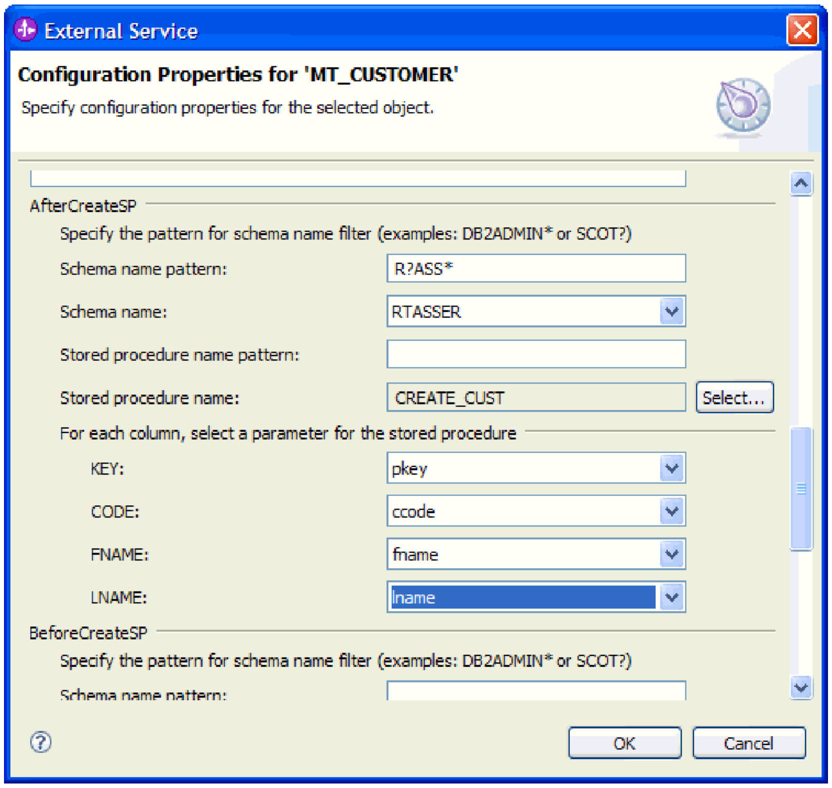
4) When the desired schema is listed, select it and click **OK**.

b. Specify the name of the stored procedure or stored function. There are two ways to specify the name.

   - Filtering the stored procedures and stored functions to search for in the database.
     1) In **Stored procedure name pattern**, type the name of the stored procedure or stored function, or type a name pattern. Use the question mark (?) character to match a single character and the asterisk (*) to match multiple characters. The name is case-sensitive.
     2) In **Stored procedure name**, select the name of the desired procedure.
   - Selecting the name from a list of all stored procedures discovered in the database.
     1) Leave the **Stored procedure name pattern** field empty.
     2) Next to **Stored procedure name**, click **Select** to open the Select window.
     3) In the Select window, select the name of the desired stored procedure or stored function. Optionally, type a case-sensitive pattern in **Value** to narrow down the list. Use the question mark (?) character to match a single character and the asterisk (*) to match multiple characters.
     4) When the desired stored procedure or stored function is listed, select it and click **OK**.

   The Configuration Properties window expands to provide an area where you configure the stored procedure. The wizard automatically generates the list of parameters by examining the stored procedure in the database.

c. For each parameter in the stored procedure (on the left), select the table column (on the right) to pass to the stored procedure in that parameter. The following figure shows a portion of the window after a stored procedure has been configured.


External Service

Configuration Properties for 'MT_CUSTOMER'

Specify configuration properties for the selected object.

AfterCreateSP

Specify the pattern for schema name filter (examples: DB2ADMIN* or SCOT?)

Schema name pattern:	R?ASS*
Schema name:	RTASSER
Stored procedure name pattern:	
Stored procedure name:	CREATE_CUST

For each column, select a parameter for the stored procedure

KEY:	pkey
CODE:	ccode
FNAME:	fname
LNAME:	lname

BeforeCreateSP

Specify the pattern for schema name filter (examples: DB2ADMIN* or SCOT?)

Schema name pattern:

7. When all fields on the window are completed, click **OK** to save the configuration of the business object. The table, view, synonym, and nickname business objects you defined are now listed in the Object Discovery and Selection window.
8. To change the configuration of an object in the **Selected objects** area, select the object name and then click the (Edit) icon.

Continue working in the Object Discovery and Selection window to select and configure other types of business objects. When you have selected and configured all business objects that you need, click **Next** to set global properties and configure wrapper business objects.

Selecting and configuring stored procedures and stored functions

Select and configure business objects corresponding to stored procedures and stored functions in the database.

Before you begin

To select and configure business objects for stored procedures or stored functions, you need to understand the structure of the data in the database and which objects the module needs to access. In particular, you need to know the parameters passed to the stored procedures or stored functions that your module needs to access.

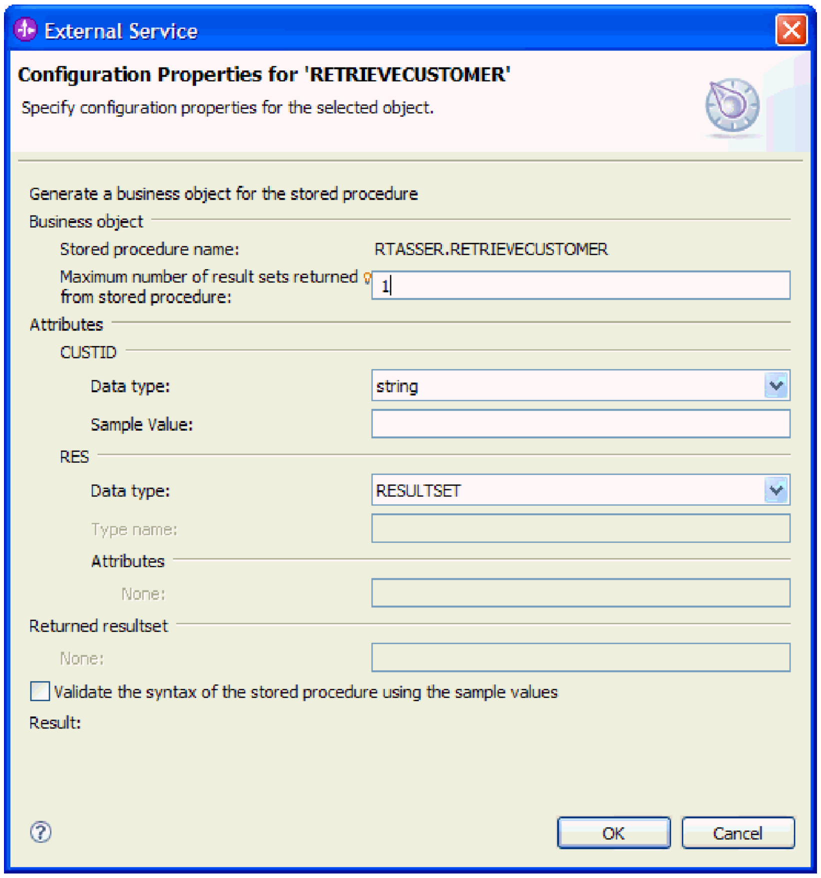
About this task

This task is performed using the external service wizard. You start in the Object Discovery and Selection window and then work in a Configuration Properties window that is specific to the business object you are configuring.

Procedure

1. In the **Discovered objects** list of the Object Discovery and Selection window, expand the node for the schema that contains the desired stored procedure or stored function, then expand the **Stored Procedures** node.
2. Optionally, you can filter database objects to make it easier to find the database objects in the tree.
   a. Click **Stored Procedures** and then click the (Edit or create filter) button, located at the top of the **Discovered objects** pane.
   b. In the Filter Properties window, type a pattern in **Object name filter**. Use the question mark (?) character to match a single character and the asterisk (*) to match multiple characters.
   c. Click **OK**.
3. Select one or more objects from the **Stored Procedure** list, and click the > (Add) button to add the object to the **Selected objects** list.

   For stored procedures that are defined in PL/SQL packages, the stored procedures are displayed in the format *SPName*(*PackageName*). For example, if the EMP_MGMT package contains the CREATE_DEPT stored procedure, the stored procedure is displayed in the list as CREATE_DEPT (EMP_MGMT). The Configuration Properties for 'object' window lists the attributes of the stored procedure business object, which include the names and data type of the parameters of the stored procedure, and information about any result sets that are returned.


External Service

Configuration Properties for 'RETRIEVECUSTOMER'

Specify configuration properties for the selected object.

Generate a business object for the stored procedure

Business object

Stored procedure name: RTASSER.RETRIEVECUSTOMER

Maximum number of result sets returned from stored procedure: 1

Attributes

CUSTID

Data type: string

Sample Value:

RES

Data type: RESULTSET

Type name:

Attributes

None:

Returned resultset

None:

☐ Validate the syntax of the stored procedure using the sample values

Result:

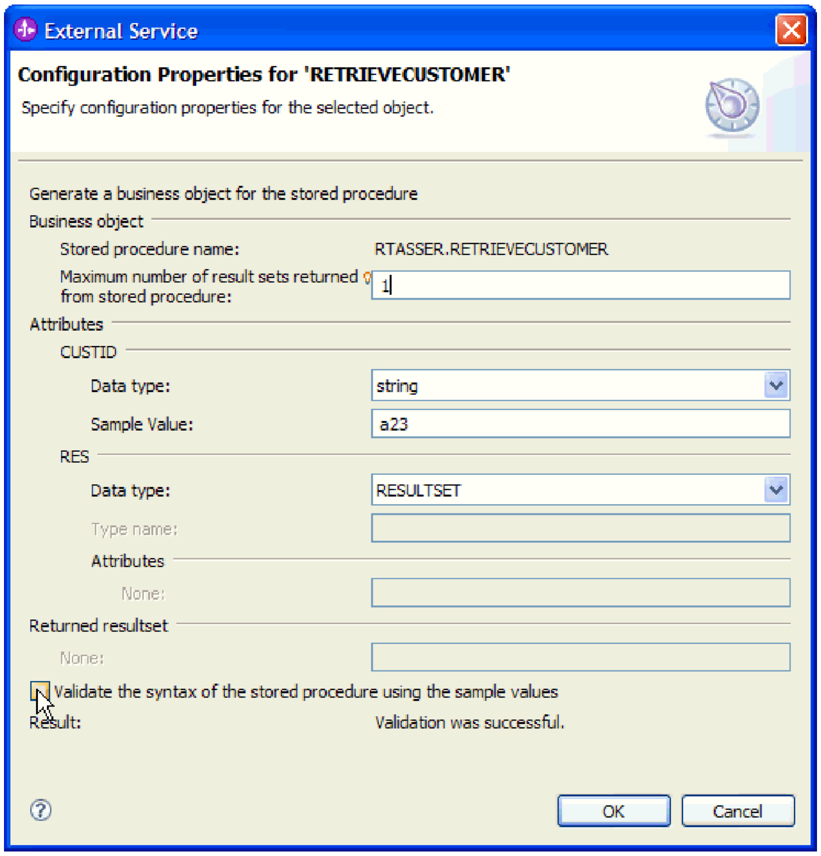
4. If the stored procedure returns any result sets, make sure that **Maximum number of results sets returned from stored procedure** lists the maximum number you expect. The wizard creates that number of result set business objects to hold the results.

   Note: For Oracle databases, make sure that the number of result sets is correct after you validate. The Oracle driver does not always return the information. If the number is not correct, set it after validating and before clicking **OK** to exit the window. After you exit the wizard, you can optionally verify the setting of the MaxNumOfRetRS application-specific information for the stored procedure business object.
5. Configure each parameter:
   a. Make sure that **Data type** displays the correct data type. For DB2 databases, the wizard can discover the data type of a parameter for you. For other types of databases, you must select the data type manually.
   b. If the attribute has a simple data type, type an actual value from your database in **Sample Value**. For example, if a parameter passes a customer's surname, you must type the surname contained in an actual customer record in the database.
6. When you have configured all attributes, click **Validate the syntax of the stored procedure using the sample values**. **Result** displays the result of the validation.

   If **Result** displays `Validation failed`, there is a problem in the information you provided. Use the error message from the database server, which follows `Validation failed`, to correct the definition. Make sure that the data type of the parameters and the sample data are correct.

   The .log file in the .metadata folder of your workspace contains additional information about the problem.

   The following figure shows the window after a stored procedure has been validated.



Configuration Properties for 'RETRIEVECUSTOMER'

Specify configuration properties for the selected object.

Generate a business object for the stored procedure

Business object

Stored procedure name: RTASSER.RETRIEVECUSTOMER

Maximum number of result sets returned from stored procedure: 1

Attributes

CUSTID

Data type: string

Sample Value: a23

RES

Data type: RESULTSET

Type name:

Attributes

None:

Returned resultset

None:

Validate the syntax of the stored procedure using the sample values

Result: Validation was successful.

7. When you see the message `Validation was successful`, click **OK** to save the definition of the stored procedure business object.

   Important: If the stored procedure or stored function returns a result set, do not click **OK** until validation succeeds. The wizard uses the results returned during validation to create business objects to hold the result. If the procedure does not validate, the adapter will not be able to return the result set at run time.
8. To change the configuration of an object in the **Selected objects** area, select the object name and then click the (Edit) icon.

Results

The business objects you configured for stored procedures and stored functions are now listed in the Object Discovery and Selection window.

Continue working in the Object Discovery and Selection window to select and configure other types of business objects. When you have selected and configured all business objects that you need, click **Next** to set global properties and configure wrapper business objects.

Selecting and configuring batch SQL business objects

Select and configure batch SQL business objects. Use a batch SQL business object to define a series of INSERT, UPDATE, and DELETE SQL statements that perform database operations.

Before you begin

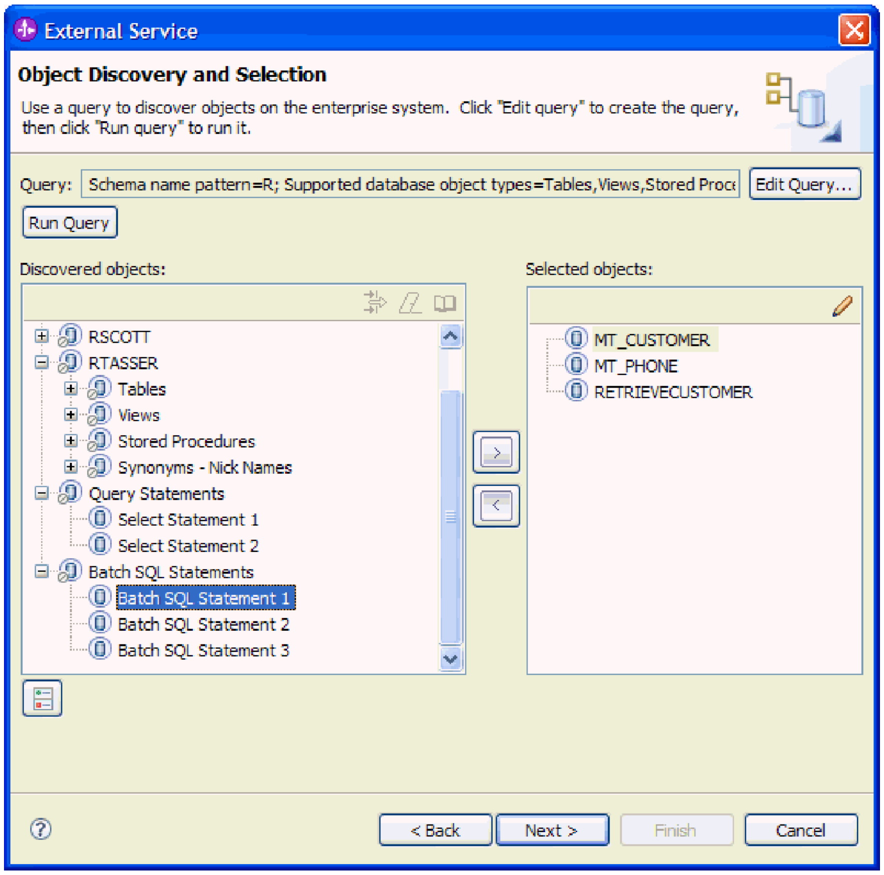
To configure batch SQL business objects, you must know the structure of the data in your database, including the tables and views. You need to know the name and data type of the columns that your SQL statements needs to process. You must also be able to write SQL INSERT, UPDATE, and DELETE statements.

About this task

This task is performed using the external service wizard. You start in the Object Discovery and Selection window and then work in a Configuration Properties window that is specific to the business object you are configuring.

Procedure

1. In the **Discovered objects** list of the Object Discovery and Selection window, expand the **Batch SQL Statements** node. This node contains an object template, named **Batch SQL Statement *n***, for each batch SQL business object you requested in the Query Properties window. For example, if you specified a count of three batch SQL business objects in that window, the **Discovered objects** list contains three object templates, as illustrated in the following figure.



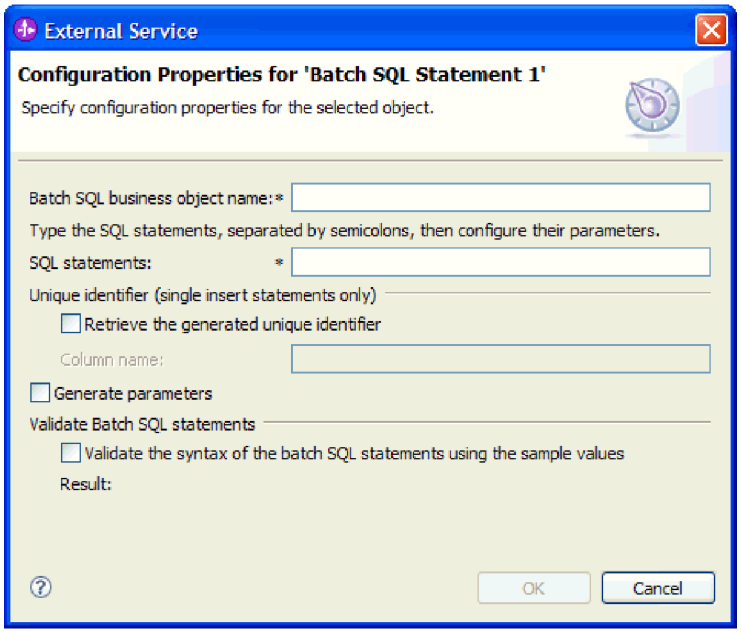
External Service

Object Discovery and Selection

Use a query to discover objects on the enterprise system. Click "Edit query" to create the query, then click "Run query" to run it.

Query: Schema name pattern=R; Supported database object types=Tables,Views,Stored Proce

2. Select one or more of the object templates and click the > (Add) button to add the object to the **Selected objects** list. The following figure shows the Configuration Properties window for a batch SQL business object, which opens when you click > (Add).


External Service

Configuration Properties for 'Batch SQL Statement 1'

Specify configuration properties for the selected object.

Batch SQL business object name:*

Type the SQL statements, separated by semicolons, then configure their parameters.

SQL statements: *

Unique identifier (single insert statements only)

Retrieve the generated unique identifier

Column name:

Generate parameters

Validate Batch SQL statements

Validate the syntax of the batch SQL statements using the sample values

Result:

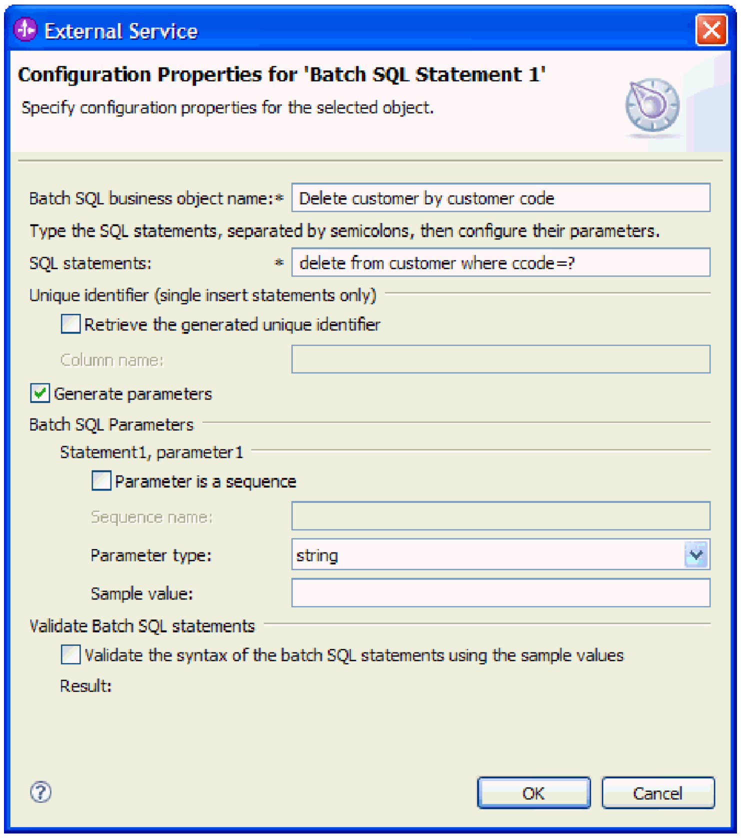
3. In **Batch SQL business object name**, type a name for the business object. The name cannot contain blanks, but may contain national language characters.
4. In **SQL statements**, type one or more SQL INSERT, UPDATE, or DELETE statements, separated by semicolons (;). Indicate each parameter in a statement with the question mark (?). The following examples demonstrate the flexibility of a batch SQL business object:
   - insert into autoid (con1) values ("Smith")
   - insert into customer (pkey, fname, lname, ccode) values (?, ?, ?, 12345)
   - update customer set fname=?, lname=? where custid=? and ccode is null
   - delete from customer where ccode like ?
   - insert into customer (pkey,ccode,fname,lname) values (?,?,?,?); delete from customer where pkey=?
5. In a DB2 or Microsoft SQL database, if you specified a single INSERT statement, you can optionally have the adapter retrieve the automatically generated unique identifier for a sequence. To configure the business object to retrieve the identifier, select **Retrieve the generated unique identifier** and then type the name of the column that contains the identifier.

   This option is valid only when you specify a single INSERT statement and the database is configured to generate an ID for the column you specify.
6. Click **Generate parameters**. The window expands to display an area where you define each parameter. This might cause the window to scroll. Expand the window for easier viewing. The areas for configuring the parameters are labeled **Statement 1, parameter 1**, **Statement *n*, parameter *m***, and so on.

   For example, suppose you specify the following SQL statements and then click **Generate parameters**: `Insert into customer (pkey,ccode,fname,lname) values(?,?,?,?); Delete from Customer where pkey=?`

   The Configuration Properties window expands to show 5 parameters. The first statement (`Insert`) has four parameters, which correspond to **Statement 1, parameter 1** through **Statement 1, parameter 4**. The second statement (`Delete`) has one parameter, **Statement 2, parameter 1**.

   The following figure shows the Configuration Properties window with one SQL statement that has one parameter.


External Service

Configuration Properties for 'Batch SQL Statement 1'

Specify configuration properties for the selected object.

Batch SQL business object name:* Delete customer by customer code

Type the SQL statements, separated by semicolons, then configure their parameters.

SQL statements: * delete from customer where ccode=?

Unique identifier (single insert statements only)

☐ Retrieve the generated unique identifier

Column name:

☑ Generate parameters

Batch SQL Parameters

Statement1, parameter1

☐ Parameter is a sequence

Sequence name:

Parameter type: string

Sample value:

Validate Batch SQL statements

☐ Validate the syntax of the batch SQL statements using the sample values

Result:

OK Cancel

7. Configure each parameter in the order you specified them in the SQL statements.
   - If the parameter is a sequence column in a DB2 or Oracle database:
     a. Click **Parameter is a sequence**.
     b. In **Sequence name**, type the name of the sequence column.

        A sequence column must be the integer data type, so **Parameter type** changes to `int`.

        No sample value is needed for a sequence column.
   - If the parameter is not a sequence column:
     a. Make sure that **Parameter is a sequence** is not selected.
     b. In **Parameter type**, select the data type of the parameter.
     c. In **Sample value**, type a sample value of the parameter. This value is used to validate that the SQL statements you entered are syntactically correct.

        For INSERT statements, you can use any value that matches the parameter's data type.

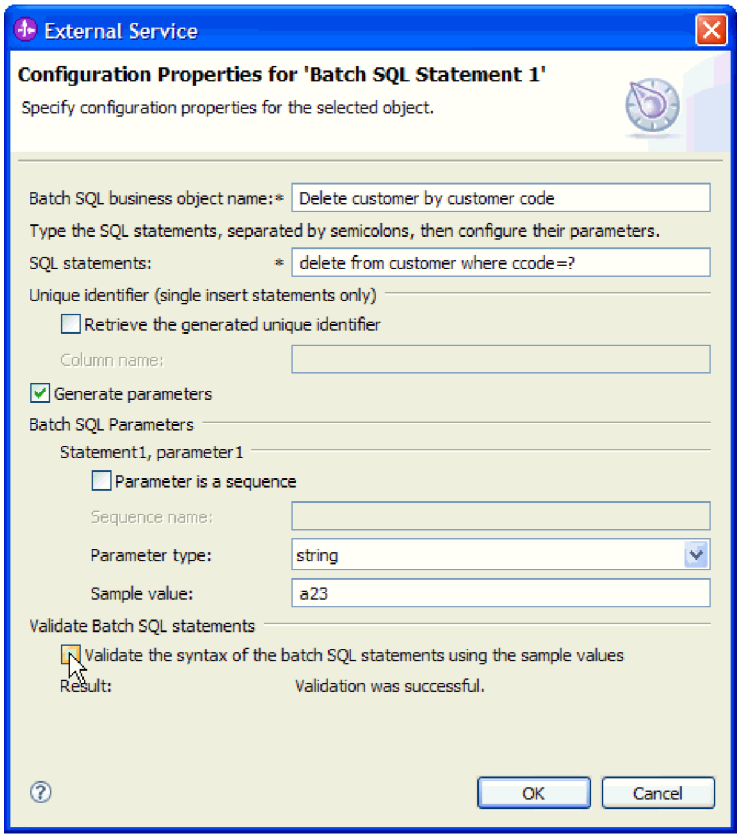
        For UPDATE and DELETE statements, you must provide a value that exists in the database. The wizard runs the statements with the sample data to get the results set, which it uses to set the attributes of the batch SQL business object. The wizard runs the statements but does not COMMIT the result, so data is not updated in or deleted from the database.

   For example, for a parameter corresponding to a column containing a customer's surname, you might select `string` as the data type and provide a sample value of `Smith`.

8. Click **Validate the syntax of the batch SQL statements using the sample values**. The **Result** line displays the result of the validation.

   If **Result** displays `Validation failed`, there is a problem in the information you provided. Use the error message from the database server, which follows `Validation failed`, to correct the definition. Check the syntax of the SQL statements, the data type of the parameters, and for UPDATE and DELETE statements, make sure the sample data exists in the database.

   The following figure shows the Configuration Properties window for a validated batch SQL business object.



External Service

Configuration Properties for 'Batch SQL Statement 1'

Specify configuration properties for the selected object.

Batch SQL business object name:* Delete customer by customer code

Type the SQL statements, separated by semicolons, then configure their parameters.

SQL statements: * delete from customer where ccode=?

Unique identifier (single insert statements only)

☐ Retrieve the generated unique identifier

Column name:

☑ Generate parameters

Batch SQL Parameters

Statement1, parameter1

☐ Parameter is a sequence

Sequence name:

Parameter type: string

Sample value: a23

Validate Batch SQL statements

☐ Validate the syntax of the batch SQL statements using the sample values

Result: Validation was successful.

9. When you see the message `Validation was successful`, click **OK** to save the definition of the batch SQL business object.

Results

The batch SQL business objects you configured are now listed in the Object Discovery and Selection window.

Continue working in the Object Discovery and Selection window to select and configure other types of business objects. When you have selected and configured all business objects that you need, click **Next** to set global properties and configure wrapper business objects.

Selecting and configuring query business objects

Select and configure query business objects for user-defined SELECT statements for use in your module.

Before you begin

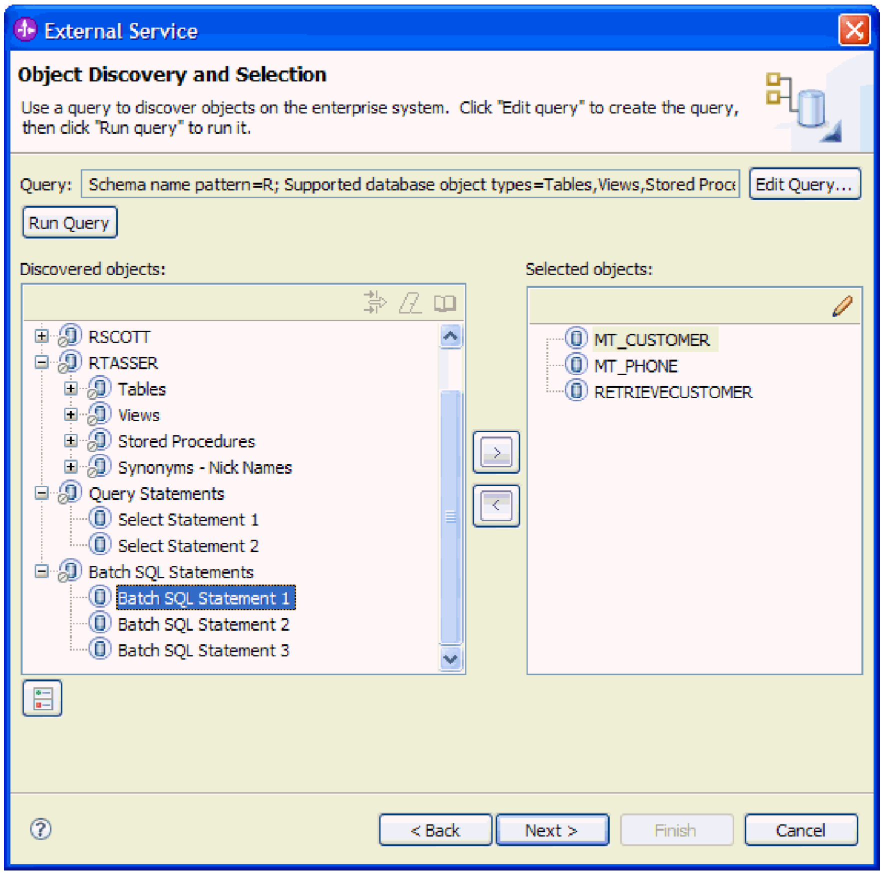
To configure query business objects, you must know the structure of the data in your database, including the tables and views. You need to know the name and data type of the columns that your module needs to access. You must also be able to write SQL SELECT statements.

About this task

This task is performed using the external service wizard. You start in the Object Discovery and Selection window and then work in a Configuration Properties window that is specific to the business object you are configuring.

Procedure

1. In the **Discovered objects** list of the Object Discovery and Selection window, expand the **Query Statements** node. This node contains an object template, named **Select Statement** *n*, for each query business object you requested in the Query Properties window. For example, if you specified a count of two query business objects in that window, the **Discovered objects** list contains two object templates, as illustrated in the following figure.


External Service

Object Discovery and Selection

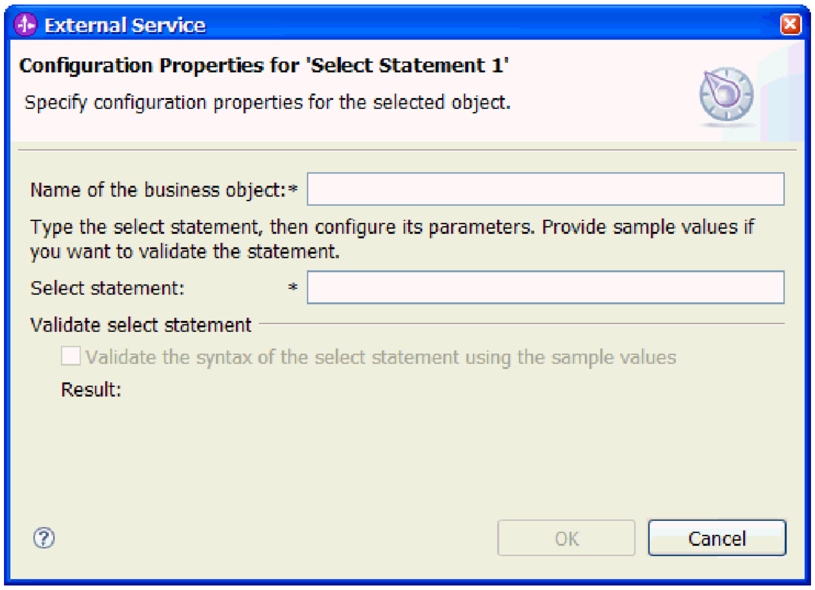
Use a query to discover objects on the enterprise system. Click "Edit query" to create the query, then click "Run query" to run it.

Query: Schema name pattern=R; Supported database object types=Tables,Views,Stored Proc

Discovered objects:
- RSCOTT
- RTASSER
  - Tables
  - Views
  - Stored Procedures
  - Synonyms - Nick Names
- Query Statements
  - Select Statement 1
  - Select Statement 2
- Batch SQL Statements
  - Batch SQL Statement 1
  - Batch SQL Statement 2
  - Batch SQL Statement 3

Selected objects:
- MT_CUSTOMER
- MT_PHONE
- RETRIEVECUSTOMER

2. Select one or more of the object templates and click the > (Add) button to add the objects to the **Selected objects** list. The following figure shows the Configuration Properties window that opens when you click > (Add) for a query business object.



Configuration Properties for 'Select Statement 1'

Specify configuration properties for the selected object.

Name of the business object:*

Type the select statement, then configure its parameters. Provide sample values if you want to validate the statement.

Select statement: *

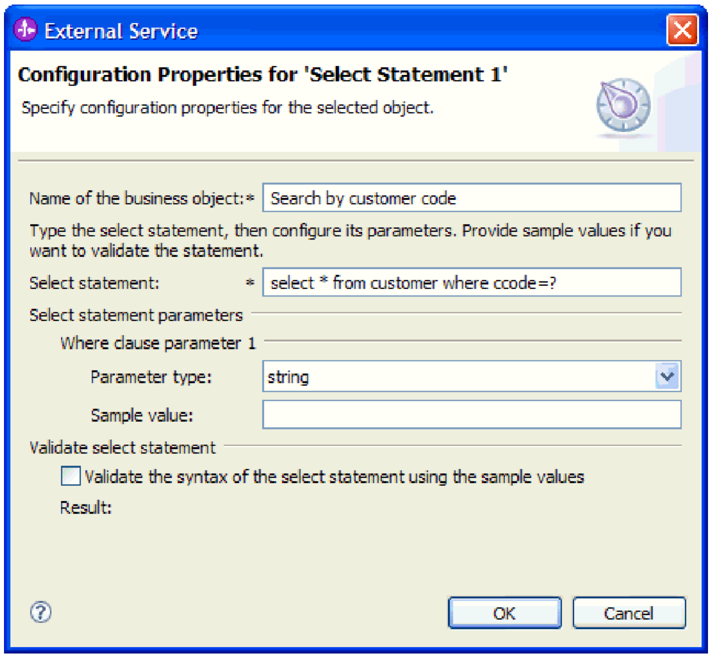
Validate select statement

Validate the syntax of the select statement using the sample values

Result:

3. In **Name of the business object**, type a name for the business object. The name can contain spaces and national language characters.
4. In **Select statement**, type the SELECT statement you want to run. Indicate each parameter with a question mark (**?**). The following sample SELECT statements illustrate the flexibility of the query business object:
   - `select * from customer where ccode=?`
   - `select * from customer where id=? and age=?`
   - `select * from customer where lname like ?`
   - `select C.pkey, C.fname, A.city from customer C, address A WHERE (C.pkey = A.custid) AND (C.fname like ?)`

   As you type each **?**, the window expands to display an area where you define the WHERE clause for that parameter. The following figure shows the Configuration Properties window for a query business object that has a single parameter.


External Service

Configuration Properties for 'Select Statement 1'

Specify configuration properties for the selected object.

Name of the business object:*	Search by customer code

Type the select statement, then configure its parameters. Provide sample values if you want to validate the statement.

Select statement: *	select * from customer where ccode=?

Select statement parameters

Where clause parameter 1

Parameter type:	string
Sample value:	

Validate select statement

☐ Validate the syntax of the select statement using the sample values

Result:

5. In **Where clause parameter *n***, provide information about each parameter in the SELECT statement.
   a. In **Parameter type**, select the data type of the parameter.
   b. In **Sample value**, type a sample value for the parameter.

   For example, for a parameter corresponding to a column containing a customer's last name, you might select `string` as the data type and provide a sample value of `Smith`.
6. Click **Validate the syntax of the select statement using the sample values**. **Result** displays the result of the validation.

   If **Result** displays `Validation failed`, there is a problem in the information you provided. Use the error message from the database server, which follows `Validation failed`, to correct the definition. Check the syntax of the SELECT statement, the data type of the parameters, and the sample data.
7. When you see the message `Validation was successful`, click **OK** to save the definition of the query business object.

Results

The query business objects you defined are now listed in the Object Discovery and Selection window.

Continue working in the Object Discovery and Selection window to select and configure other types of business objects. When you have selected and configured all business objects that you need, click **Next** to set global properties and configure wrapper business objects.

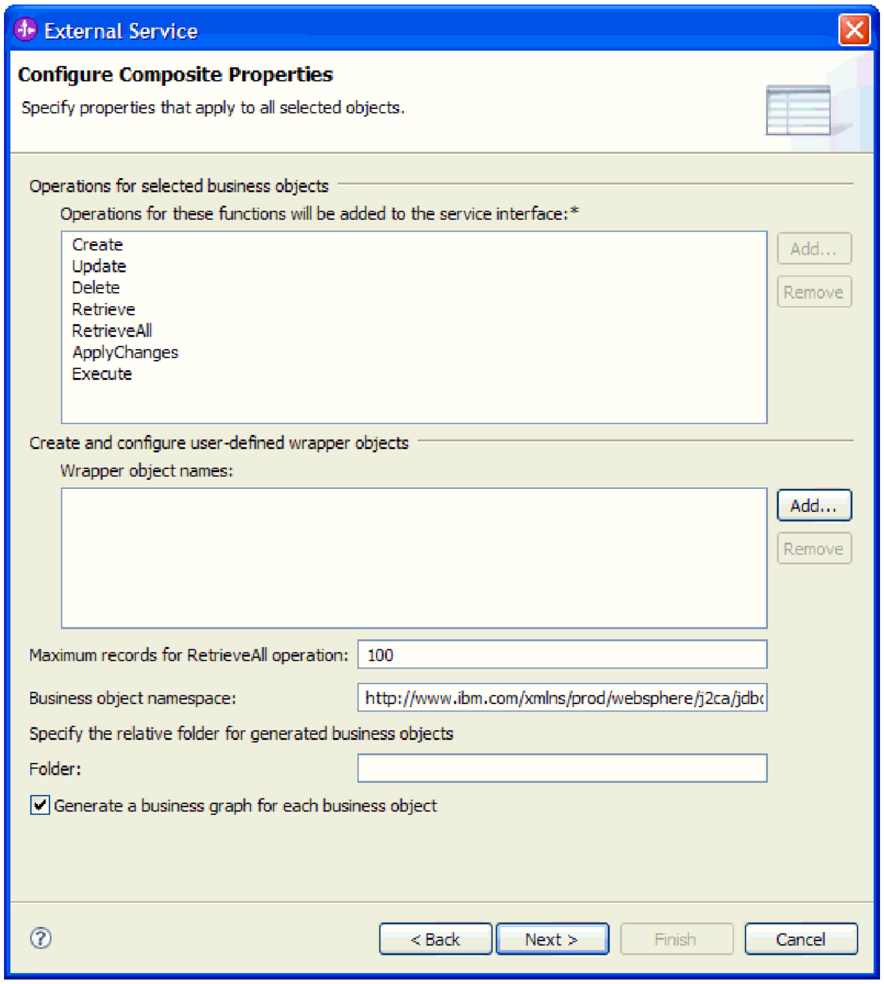
Setting global properties for operations and creating wrapper business objects

After you select database objects in the external service wizard, you need to define business objects for wrappers, and specify properties that apply to all business objects.

Procedure

1. When the **Selected objects** list in the Object Discovery and Selection window contains all of the business objects you want to use in your application, except for wrapper business objects, click **Next**.
2. In the Configure Composite Properties window, review the list of operations.

   This window lists all of the operations that the adapter supports for the outbound services for all business objects that you selected on the previous window. Not all operations are supported by each business object. For example, query business objects support only the RetrieveAll operation. Stored procedure and batch SQL business objects support only the Execute operation.



External Service

Configure Composite Properties

Specify properties that apply to all selected objects.

Operations for selected business objects

Operations for these functions will be added to the service interface:*

Create
Update
Delete
Retrieve
RetrieveAll
ApplyChanges
Execute

Create and configure user-defined wrapper objects

Wrapper object names:

Maximum records for RetrieveAll operation: 100

Business object namespace: http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc

Specify the relative folder for generated business objects

Folder:

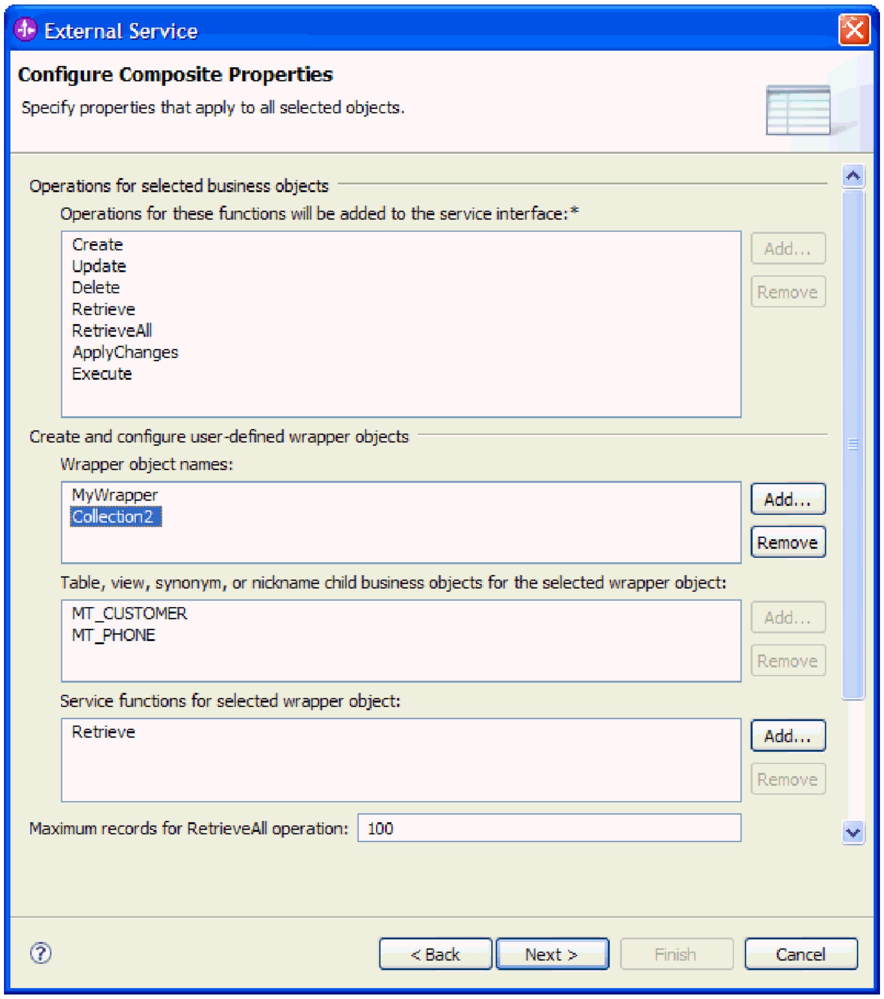
[✓] Generate a business graph for each business object

3. To remove an operation that you do not need, select the operation name and click **Remove**. If you change your mind, click **Add** and restore a removed operation.
4. To create a wrapper business object:
   a. In the **Wrapper object names** area, click **Add**.
   b. In the **Add** window, type the name of your wrapper business object and then click **OK**. Do not use spaces. The name can contain national language characters.
   c. In **Table, view, synonym, or nickname child objects for the selected wrapper**, click **Add**.
   d. In the Add window, select one or more business objects that you want to include in the wrapper and then click **OK**.

e. In **Service functions for selected wrapper object**, click **Add**.

f. In the Add window, select one or more operations that you want to perform on the wrapper object and then click **OK**. The RetrieveAll and ApplyChanges operations are not listed because they do not apply to wrapper business objects.

g. Repeat this procedure for each wrapper business object you want to create. The following figure shows the Configure Composite Properties window with two wrapper business objects defined. The window shows the properties of one wrapper business object at a time.


External Service

Configure Composite Properties

Specify properties that apply to all selected objects.

Operations for selected business objects

Operations for these functions will be added to the service interface:*

Create
Update
Delete
Retrieve
RetrieveAll
ApplyChanges
Execute

Create and configure user-defined wrapper objects

Wrapper object names:

MyWrapper
Collection2

Table, view, synonym, or nickname child business objects for the selected wrapper object:

MT_CUSTOMER
MT_PHONE

Service functions for selected wrapper object:

Retrieve

Maximum records for RetrieveAll operation: 100

5. In **Maximum records for RetrieveAll operations**, type the upper limit on the number of records to retrieve for a RetrieveAll operation. The default value is 100. For more information about this property, see “Maximum records for RetrieveAll operation” on page 196.

6. In **Business object namespace**, accept the default namespace or type the full name of another namespace.

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

7. Optionally, in **Folder**, type the relative path to the folder where the generated business objects are to be stored.

8. If you want a business graph to be created for each business object, click **Generate a business graph for each business object**. Business graphs are needed only in these situations:

- If you need to use the ApplyChanges operation
- When adding business objects to a module created with a version of WebSphere Integration Developer earlier than version 6.1.0.

  Note: You must select this option if you are adding business objects to a module that was created with an earlier version of WebSphere Integration Developer. Otherwise, you must rewire your interface.

9. Click **Next**.

Results

You have created wrapper business objects and provided information that applies to all business objects in the module.

What to do next

Continue working in the wizard. The next step is to specify deployment information to use at run time and information for saving the service as a module.

Setting deployment properties and generating the service

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

About this task

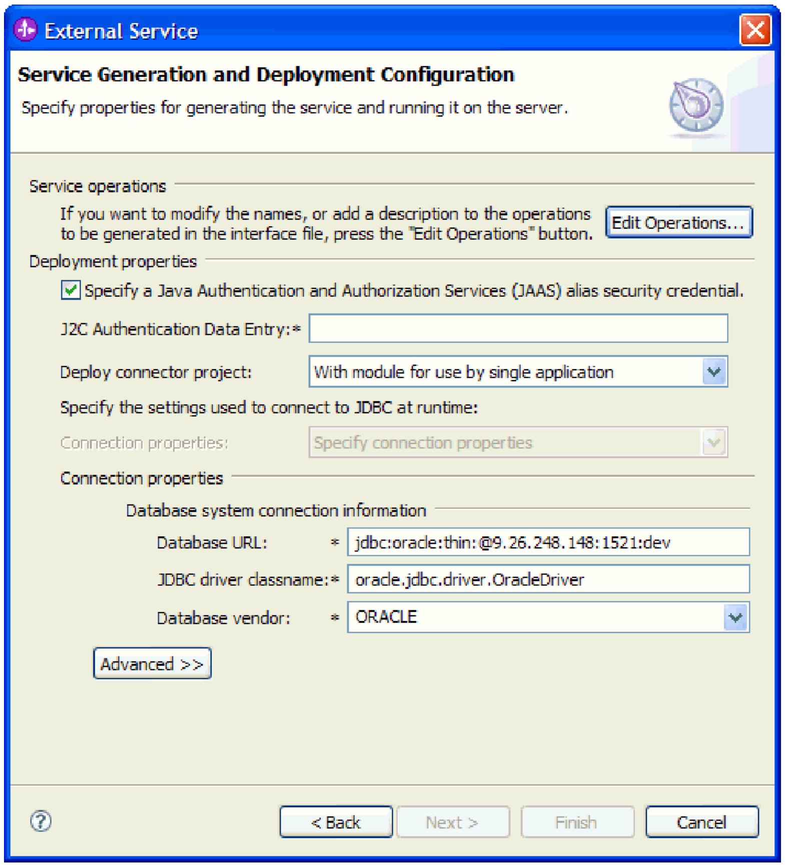
This task is performed using the Service Generation and Deployment Configuration and Service Location Properties windows of the external service wizard.

The connection properties in this task are initialized to the values that the wizard used to connect to the database. To configure the module to use other values, change the values here. For example, to use the IBM Toolkit for Java native driver at run time on i5/OS, set the driver information here.

Procedure

1. In the Service Generation and Deployment Configuration window, click **Edit Operations** to review the names of or add a description for the operations for

the business objects you are creating.


External Service

Service Generation and Deployment Configuration

Specify properties for generating the service and running it on the server.

Service operations

If you want to modify the names, or add a description to the operations to be generated in the interface file, press the "Edit Operations" button.

Deployment properties

[✓] Specify a Java Authentication and Authorization Services (JAAS) alias security credential.

J2C Authentication Data Entry:*

Deploy connector project: With module for use by single application

Specify the settings used to connect to JDBC at runtime:

Connection properties: Specify connection properties

Connection properties

Database system connection information

Database URL: * jdbc:oracle:thin:@9.26.248.148:1521:dev

JDBC driver classname:* oracle.jdbc.driver.OracleDriver

Database vendor: * ORACLE

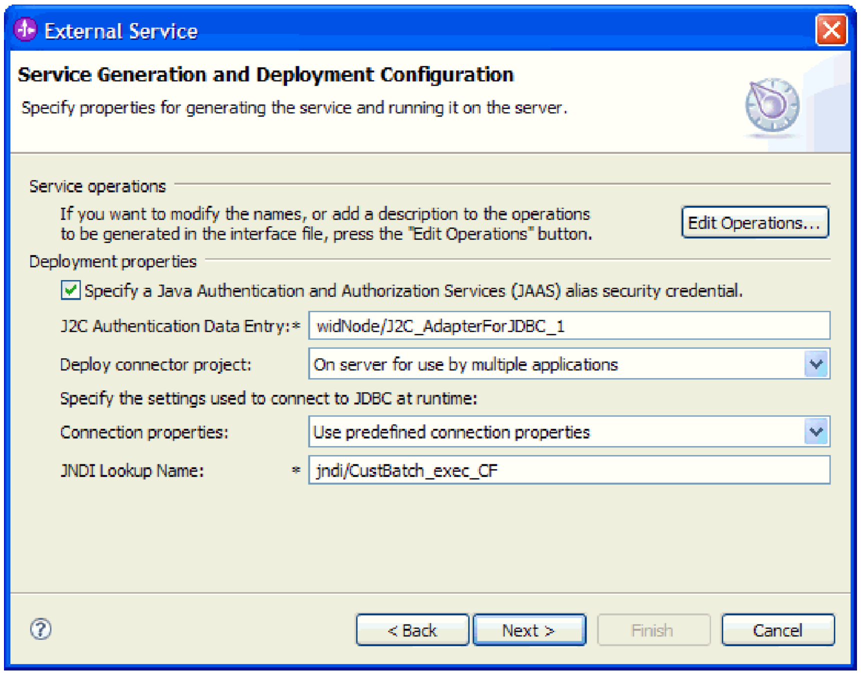
2. Specify how you want the adapter to get the database user name and password at run time.
   - To use a J2C authentication alias, select **Specify a Java Authentication and Authorization Services (JAAS) alias security credential** and type the name of the alias in **J2C Authentication Data Entry**.

     You can specify an existing authentication alias or create one at any time before deploying the module. The name is case-sensitive and includes the node name.
   - To use an existing data source on the server:
     a. Clear **Specify a Java Authentication and Authorization Services (JAAS) alias security credential**.
     b. Click **Advanced**.
     c. Expand **Alternate ways to specify connection information**.
     d. Complete one of these sets of fields:
        – **Datasource JNDI name**
        – **XA datasource name** and **XA database name**
   - To specify the database user name and password to be saved in the adapter properties:
     a. Clear **Specify a Java Authentication and Authorization Services (JAAS) alias security credential**.
     b. Click **Advanced**.
     c. Under **Database system connection properties**, type the **User name** and **Password**.

Note: When you specify the password here, it is saved as clear text in an adapter property, which unauthorized users might be able to see.

3. In **Deploy connector project**, specify whether to include the adapter files in the module. Choose one of the following values:
   - **With module for use by single application**. With the adapter files embedded in the module, you can deploy the module to any application server. Use an embedded adapter when you have a single module using the adapter or if multiple modules need to run different versions of the adapter. Using an embedded adapter enables you to upgrade the adapter in a single module without the risk of destabilizing other modules by changing their adapter version.
   - **On server for use by multiple applications**. If you do not include the adapter files in a module, you must install them as a stand-alone adapter on each application server where you want to run the module. Use a stand-alone adapter when multiple modules can use the same version of the adapter and you want to administer the adapter in a central location. A stand-alone adapter can also reduce the resources required by running a single adapter instance for multiple modules.
4. If you specified **On server for use by multiple adapters** in the previous step, specify how you want to specify the connection properties.
   - If you manually created and configured a managed connection factory or activation specification on the server or if you have already deployed an application that connects to the same database using the same managed connection factory or activation specification properties, you can reuse the managed connection factory or activation specification by specifying the name of its Java Naming and Directory Interface (JNDI) data source:
     a. In **Connection properties**, select **Use predefined connection properties**.
     b. In **JNDI Lookup Name**, type the name of the JNDI data source for an existing managed connection factory or activation specification.

        The following figure shows typical settings for reusing a managed connection factory or activation specification for a stand-alone deployment of the adapter.



Service Generation and Deployment Configuration

Specify properties for generating the service and running it on the server.

Service operations

If you want to modify the names, or add a description to the operations to be generated in the interface file, press the "Edit Operations" button.

Deployment properties

✓ Specify a Java Authentication and Authorization Services (JAAS) alias security credential.

J2C Authentication Data Entry:* widNode/J2C_AdapterForJDBC_1

Deploy connector project: On server for use by multiple applications

Specify the settings used to connect to JDBC at runtime:

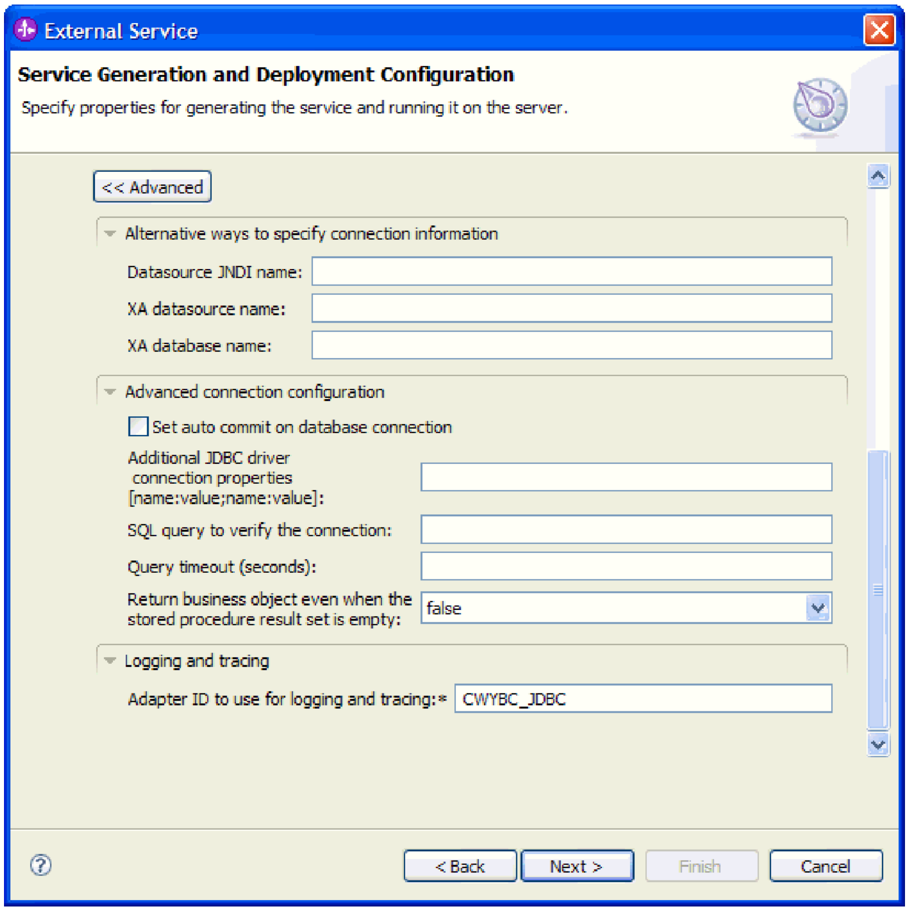
Connection properties: Use predefined connection properties

JNDI Lookup Name: * jndi/CustBatch_exec_CF

c. Click **Next** to complete this task.

- If this is the first application that connects to the database with a specific user name and password, or if you want to administer the user name and password separately from other applications, select **Specify connection properties**.

5. Review and, if necessary, change the values of the required connection properties. The fields are initialized with the connection information you specified when you started the wizard. You can change the values to specify a different user name and password at run time. You can also connect to an alternate database, although the schema names must be the same in both databases. The format of the connection properties are database-specific. For more information about the properties, see “Managed connection factory properties” on page 187.
6. Optionally, specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties. The following figure shows the advanced properties on the Service Generation and Deployment Configuration window.


External Service

Service Generation and Deployment Configuration

Specify properties for generating the service and running it on the server.

<< Advanced

Alternative ways to specify connection information

Datasource JNDI name:

XA datasource name:

XA database name:

Advanced connection configuration

☐ Set auto commit on database connection

Additional JDBC driver connection properties [name:value;name:value]:

SQL query to verify the connection:

Query timeout (seconds):

Return business object even when the stored procedure result set is empty: false

Logging and tracing

Adapter ID to use for logging and tracing:* CWYBC_JDBC

- The properties specified in **Alternate ways to specify connection information** were described in a previous step.
- **Advanced connection configuration**

  a. If you want to turn AUTOCOMMIT on for the database, select **Set auto commit on database connection**. See more information about the property in “Auto commit (AutoCommit)” on page 189.

b. Set **Additional JDBC driver connection properties**. See more information about the property in "Additional JDBC driver connection properties (DriverConnectionProperties)" on page 189.

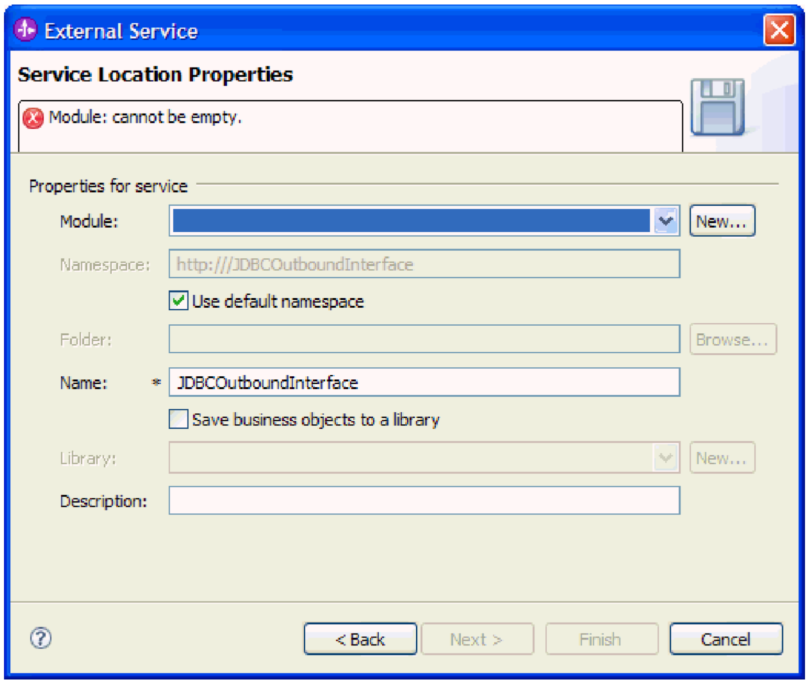
c. In **Query timeout**, type the length of time, in seconds, that the adapter should wait for a response to a database query. See more information about the property in "Query timeout (QueryTimeOut)" on page 193.

d. Set **SQL query to verify the connection**, See more information about the property in "SQL query to verify the connection (PingQuery)" on page 194.

e. Set **Return business object even when the stored procedure result set is empty**. See more information about the property in "Return business object even when the stored procedure result set is empty (ReturnDummyBOForSP)" on page 194.

- **Logging and tracing**
  - In **Adapter ID to use for logging and tracing**, type a different adapter ID. See more information about the property in "Adapter ID to use for logging and tracing (AdapterID)" on page 188.

7. Click **Next**. The Service Location Properties window is displayed.



Module: cannot be empty.

Module:
Namespace: http:///JDBCOutboundInterface
Folder:
Name: * JDBCOutboundInterface
Library:
Description:

8. In the Service Location Properties window, specify the name of the module you want to create. This can be a new or existing module.
   - If the desired module name appears in the **Module** list, select its name.

     Important: If the module contains an interface or business object with the same name as any you are now configuring, the original interface or business object in the module will be replaced by the new version.
   - Otherwise, create a new module:

     a. Click **New**.

     b. In the Integration Project window, select **Create a module project** and click **Next**.

     c. In the Module window, type a name for the module. For example, JDBCOutboundModule.

   d. If you want the service description files (the .import and .wsdl files) to be located in the default folder in the module, leave **Use default location** selected. If you want to specify a different folder in the module, clear the option and then click **Browse** to specify a different folder in **Location**.
   e. If you want the module to be automatically opened in the assembly diagram in WebSphere Integration Developer when the wizard closes, select **Open module assembly diagram**. Otherwise, clear this option.
   f. Click **Finish** to create the new module.
9. Specify the namespace you want to use for your business objects.
   - If you want the business objects in the module to use the default derived namespace, leave **Use default namespace** selected.
   - To specify a different namespace, clear the option and type a different value in **Namespace**.
10. Optionally, specify the folder within the new module where the service description will be saved. In **Folder**, type the folder name or browse to an existing folder. If you do not specify a folder name, the artifacts (the import, XSD, and WSDL files) are stored in the root folder of the module, that is, the folder with the module name.
11. In **Name**, accept the default import name or type a different name.
12. Optionally, if you want to save the business objects in a library where they can be used by other modules, select **Save business objects to a library** and specify the location of the library in **Library**.
13. Optionally, in **Description**, type a descriptive comment about the module.
14. When you are finished setting properties, click **Finish**.
15. If the Model Changed window is displayed, click **Yes**.

Results

The wizard exits. The module is created in the project and artifacts are generated.

What to do next

In some instances, you might need to use the assembly editor to complete the configuration. Then you can test or deploy your module.

Completing the configuration

In some situations, manual configuration steps are needed to complete the configuration of your business objects.

About this task

2 Perform this task when you need to customize the artifacts generated by the
2 wizard. You might do this in the following situations:

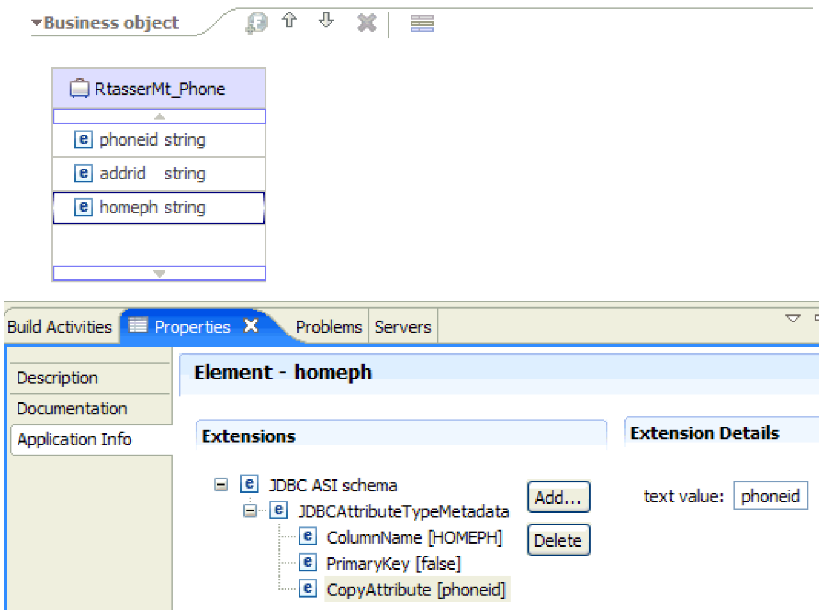
- 2 To set the CopyAttribute parameter for a column so that its value is set to the
2 same value as another column.
- To add or remove attributes from a business object. For example, you can simplify your business object design by removing the simple attribute corresponding to any database column that you do not need to reference.
- To configure additional parents for a table business object that has multiple parents. The wizard configures only one parent for a table business object.

This topic provides detailed instructions for setting the CopyAttribute parameter to a table business object. Other changes to business object structures can be accomplished using similar techniques.

The CopyAttribute parameter is contained in the properties of the attribute for the column that you want to populate with values and application-specific information from another column. For example, if you want the `contact` column of a new row in the table to contain the same value as the `email` column, set the CopyAttribute parameter of the `contact` attribute to `email`. You use the assembly editor in WebSphere Integration Developer to set the value.

Procedure

1. In the Business Integration perspective in WebSphere Integration Developer, expand the module name, expand **Data Type**, and then locate the table business object. The business object name is the name of the database schema plus the name of the database table. An optional namespace might be included at the beginning of the name.
2. Right-click the business object name and select **Open**. The assembly editor displays the business object, which has a field for each column.
3. In the assembly editor, select the column you want to set to match another column.
4. In the Properties view, select Application Info. If the Properties view is not visible, right-click the column name and click **Show in Properties**.
5. Expand **JDBC ASI schema**, and then expand **JDBCAttributeTypeMetadata**.
6. Right-click **JDBCAttributeTypeMetadata** and then select **New** → **jdbcasi:CopyAttribute**.
7. Select the
8. In the Extension Details area, set the text value to the name of the column that contains the information to copy. The column can be in the current business object or its parent business object. To copy from a column in the current business object, set the value to the column name, for example, `phoneid`. To copy from a column in the parent business object, prefix the column name with two periods (..), for example, `..phone`. The following figure shows the assembly editor showing the CopyAttribute property set to a column in the current table.



Element - homeph

JDBC ASI schema
JDBCAttributeTypeMetadata
ColumnName [HOMEPH]
PrimaryKey [false]
CopyAttribute [phoneid]

text value: phoneid

2 **Results**

2 The business object is configured to use the CopyAttribute property to set the
2 business object attribute for a database column using information in another
2 column.

What to do next

You can now test and deploy the module.

Configuring the module for inbound processing

To configure a module to use the adapter for inbound processing, use the external service wizard in WebSphere Integration Developer to find and select business objects and services from the database, and to generate business object definitions and related artifacts.

Discovering database objects

After configuring the connection properties, run a query to search for database objects. Browse the tree of discovered objects to understand the structure of objects in the database and use filters to display only the database objects you want to see.

Before you begin

You must know the data requirement of the program that needs to access the database. For example, you need the following information about your database:

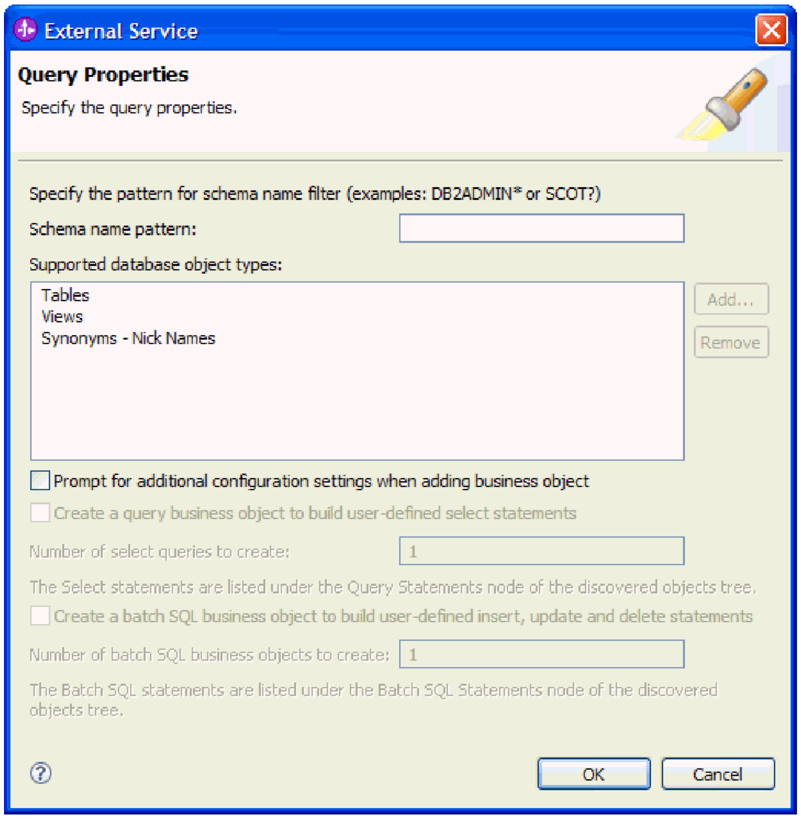
- Which schemas your module needs to access
- What type of database objects you need to access in those schemas

About this task

This task starts in the Object Discovery and Selection window of the external service wizard.

Procedure

1. In the Object Discovery and Selection window, click **Edit Query**. The Query Properties window is displayed.


External Service

Query Properties

Specify the query properties.

Specify the pattern for schema name filter (examples: DB2ADMIN* or SCOT?)

Schema name pattern:

Supported database object types:

Tables
Views
Synonyms - Nick Names

Add... Remove

☐ Prompt for additional configuration settings when adding business object

☐ Create a query business object to build user-defined select statements

Number of select queries to create: 1

The Select statements are listed under the Query Statements node of the discovered objects tree.

☐ Create a batch SQL business object to build user-defined insert, update and delete statements

Number of batch SQL business objects to create: 1

The Batch SQL statements are listed under the Batch SQL Statements node of the discovered objects tree.

OK Cancel

You might see the unavailable options **Create a query business object to build user-defined select statements** and **Create a batch SQL business object to build user-defined insert, update, and delete statements**. These options are available only for outbound processing.

Use the Query Properties window to perform the following tasks:

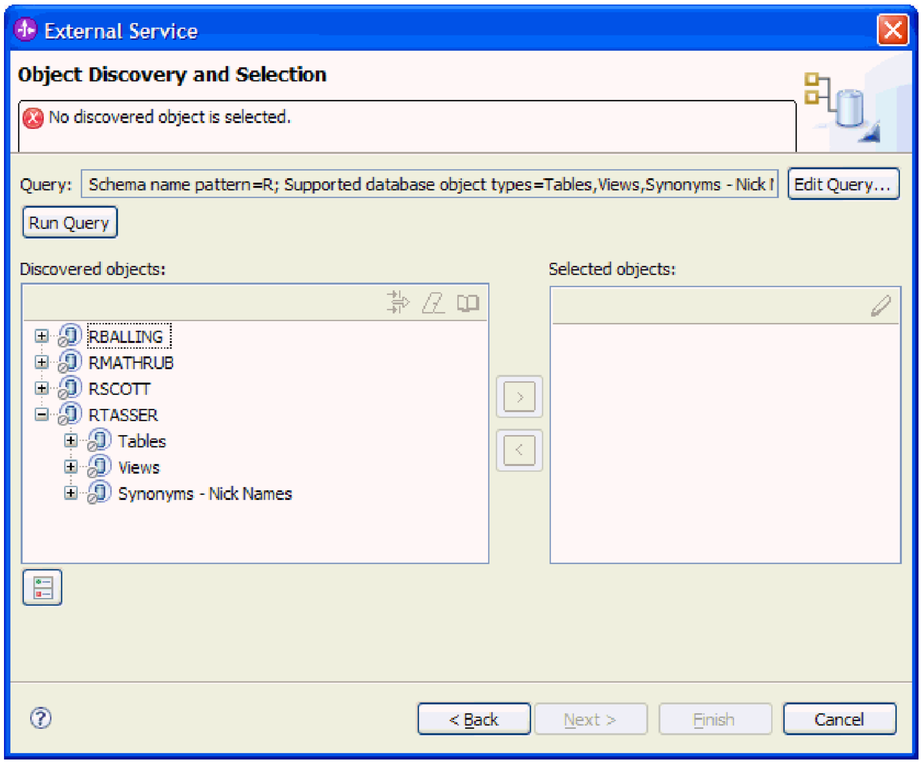
- Reduce the search time by searching a subset of database schemas
- Omit one or more types of database objects from the search
- Make the wizard prompt you for application-specific information that cannot be automatically determined based on information in the database

2. To limit the number of database schemas that are retrieved, type the name of the schema or a name pattern in **Schema name pattern**. Use the question mark (?) character to match a single character and the asterisk (*) to match multiple characters. Only schemas that start with that string or match that pattern are displayed when you run the query. If you do not specify a schema name pattern, all schemas in the database are displayed. Using a filter can speed up the discovery process if your database contains many schemas.
3. To omit one or more types of objects from the search, select the types of objects that you want to omit (tables, views, and synonyms or nicknames) in **Supported database object types**, and then click **Remove**. If you change your mind, click **Add** to add the object type back. If your database contains object types that you do not need to access, omitting them can speed up the discovery process.
4. Select **Prompt for additional configuration settings when adding business object**. Then, whenever you add a database object to the list of business objects to create, the wizard automatically prompts you for all user-configurable application-specific information for the object. For example, if you select this option, the wizard guides you through the process of building a simple

parent-child hierarchy of business objects. If you need hierarchy in which a table business object has two attributes which are referring to attributes in two different tables (that is, it has two parent business objects), you will need to complete the configuration in the assembly editor, a tool launched from WebSphere Integration Developer.

Important: If you do not select this option, the wizard prompts only for required information. You must complete the configuration of the business objects using the assembly editor.

5. Click **OK** to save your changes to the query.
6. In the Object Discovery and Selection window, click **Run Query** to use the query to discover database objects. The result of running a typical query are shown in the following figure.


External Service

Object Discovery and Selection

No discovered object is selected.

Query: Schema name pattern=R; Supported database object types=Tables,Views,Synonyms - Nick N

Discovered objects:

- RBALLING
- RMATHRUB
- RSCOTT
- RTASSER
  - Tables
  - Views
  - Synonyms - Nick Names

Selected objects:

The **Discovered objects** pane lists the objects that were discovered. The tables, views, and synonyms/nicknames are sorted by schema name.

7. In the **Discovered objects** list, click **+** (the plus sign) to expand a schema node and the **Tables**, **Views**, and **Synonyms - Nicknames** nodes underneath it to see the database objects discovered by the wizard.

Results

The wizard has discovered the database objects you can access using the adapter.

What to do next

Continue working in the external service wizard. The next step is to select the objects you want to use in your module, configure each business object, and create hierarchies of business objects.

Selecting and configuring business objects

Using the list of database objects discovered by the external service wizard, and the query and batch SQL object templates you specified, continue using the wizard to select the database objects that you need to access in your module. Then provide configuration information for your new business objects.

About this task

The Object Discovery and Selection window allows you to select and configure objects in any order, with the single exception that you must select and configure a parent table before you can select and configure its child tables. Aside from this restriction, you have the flexibility to add objects individually or to add them several at a time. You can mix objects from the various nodes of the **Discovered objects** list. For example, you can select several table and view objects and a stored procedure object, and add them at the same time.

The high-level flow of selecting and configuring business objects is as follows:

1. You select one or more objects in the **Discovered objects** list of the Object Discovery and Selection window.
2. You click the > (Add) button.
3. The wizard opens the Configuration Properties window.
   - If you selected a single object, a single Configuration Properties window is displayed.

     You complete that window, specifying any user-configurable attributes and other information that the wizard cannot discover by examining the database, and click **OK** to save the configuration.
   - If you selected multiple objects, a Configuration Properties notebook is displayed, with one page for each object selected.

     You click on the name of each object in turn. The notebook displays the same information you would see had you selected this object individually.

     Important: Do not click **OK** on the notebook until you complete the configuration pages for all of the objects. The wizard will not close the notebook until you provide all of the required fields, but you can close the notebook before providing optional fields. If you do not configure the optional fields in the wizard, you must use the business object editor to configure them after exiting the wizard.
4. The wizard adds the configured object to the **Selected objects** list.

As long as you do not exit from the wizard, you can work iteratively to select and configure the business objects you need in your module. However, you cannot use the wizard to add objects to an existing module, so be careful to understand the requirements of the program that uses the business objects before you start the wizard.

Selecting and configuring tables, views, and synonyms or nicknames

Select and configure business objects for tables, views, and synonyms or nicknames for use in your module. For inbound processing, these are the business objects that are delivered in events.

Before you begin

To select and configure business objects for tables, views, and synonyms or nicknames, you need to understand the structure of the data in the database and which database objects the module needs to access. This includes the following types of information:

- The structure of the tables, views, and synonyms or nicknames, including columns and column attributes such as data type
- The relationships between the tables, including the cardinality and ownership of parent-child relationships

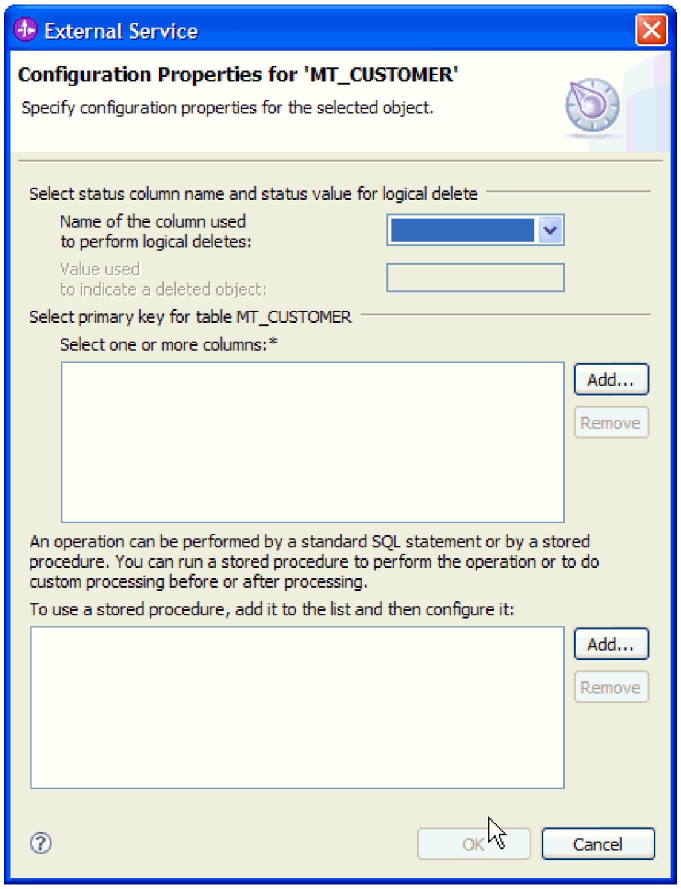
About this task

This task is performed using the external service wizard. You start in the Object Discovery and Selection window and then work in a Configuration Properties window that is specific to the business object you are configuring.

Procedure

1. In the **Discovered objects** list of the Object Discovery and Selection window, select one or more tables, views, or synonym, and click the **>** (Add) button to add the object to the **Selected objects** list.

   The following pair of figures shows a typical Configuration Properties window for a table, view, synonym, or nickname business object. The following figure shows a typical window for the first table or group of tables that you select.



External Service

Configuration Properties for 'MT_CUSTOMER'

Specify configuration properties for the selected object.

Select status column name and status value for logical delete

Name of the column used to perform logical deletes:

Value used to indicate a deleted object:

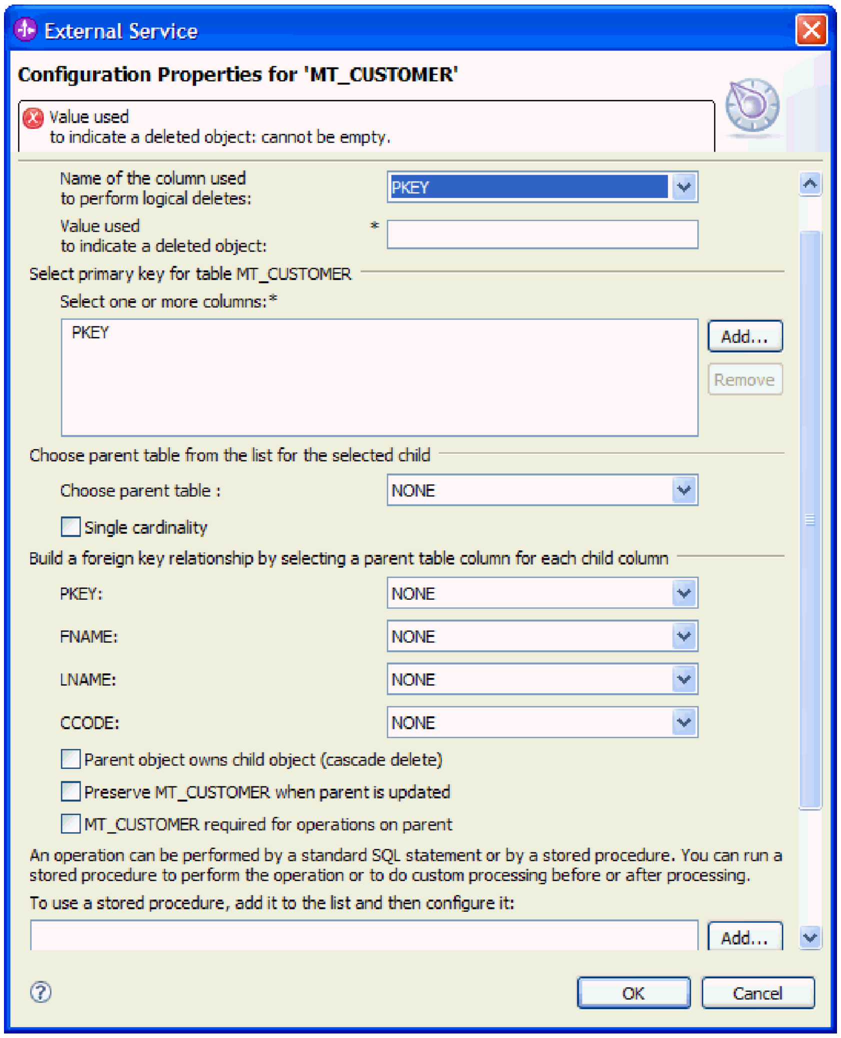
Select primary key for table MT_CUSTOMER

Select one or more columns:*

An operation can be performed by a standard SQL statement or by a stored procedure. You can run a stored procedure to perform the operation or to do custom processing before or after processing.

To use a stored procedure, add it to the list and then configure it:

   The following figure shows a typical window for subsequent tables. After you select and configure at least one table, the Configuration Properties window for subsequent tables displays an area where you can optionally define a parent-child hierarchy between tables.


External Service

Configuration Properties for 'MT_CUSTOMER'

Value used
to indicate a deleted object: cannot be empty.

Name of the column used to perform logical deletes: PKEY

Value used to indicate a deleted object: *

Select primary key for table MT_CUSTOMER

Select one or more columns:* PKEY

Add... Remove

Choose parent table from the list for the selected child

Choose parent table : NONE

Single cardinality

Build a foreign key relationship by selecting a parent table column for each child column

PKEY: NONE

FNAME: NONE

LNAME: NONE

CCODE: NONE

Parent object owns child object (cascade delete)

Preserve MT_CUSTOMER when parent is updated

MT_CUSTOMER required for operations on parent

An operation can be performed by a standard SQL statement or by a stored procedure. You can run a stored procedure to perform the operation or to do custom processing before or after processing.

To use a stored procedure, add it to the list and then configure it:

Add...

OK Cancel

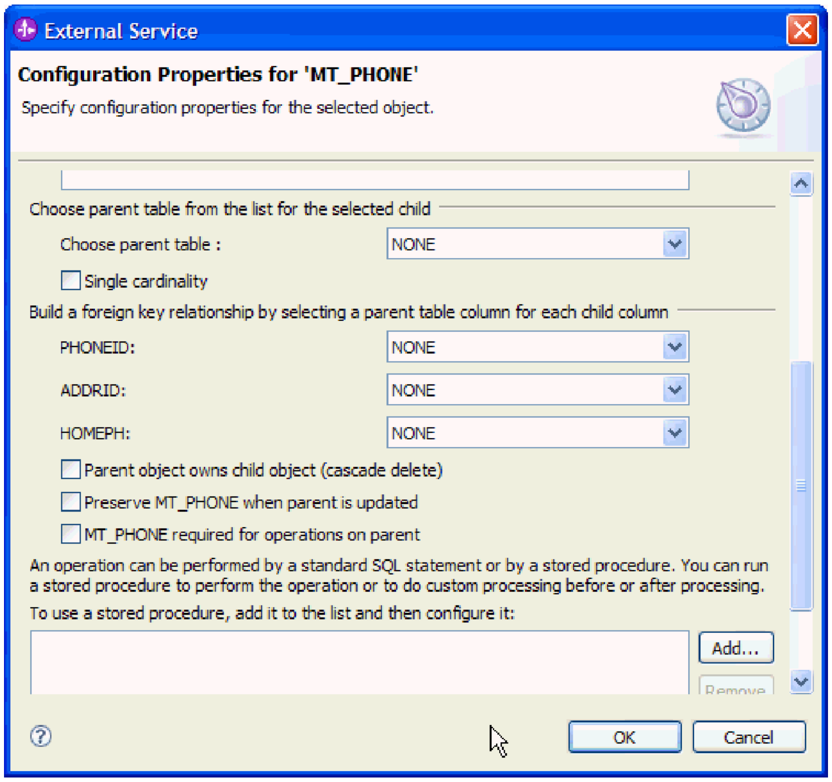
As you configure the object, choices that require advanced configuration might cause additional fields to be displayed in this window. This might cause the window to scroll. Be sure that you examine all fields on the window before clicking **OK**.

2. If the table has a column that is used to indicate logical deletes:
   a. Select the column name in **Name of the column used to perform logical deletes**.
   b. In **Value used to indicate a deleted object**, type the value that indicates that a row is logically deleted. You can get this value from your database administrator.
3. The **Select primary key for table** ***table_name*** area is displayed only when the database table does not have a column designated as the primary key. Each table business object must have a primary key, even if the associated database table does not have a key. If the primary key is defined in the database, this section of the window is not displayed.

   In **Select primary key for table** ***table_name***, click **Add**, select the column to be used as the primary key for the table business object, and then click **OK**. If the table has a composite key, you can select multiple columns.
4. Optionally, define a parent-child relationship between business objects.

Important: To build a parent-child hierarchy, configure the parent table first, and then return to the Object Discovery and Selection window to select and configure the child tables.

Configure the parent-child relationship using the area of the Configuration Properties window shown in the following figure. These fields are not displayed for the first table you configure.


External Service

Configuration Properties for 'MT_PHONE'

Specify configuration properties for the selected object.

Choose parent table from the list for the selected child

Choose parent table : NONE

☐ Single cardinality

Build a foreign key relationship by selecting a parent table column for each child column

PHONEID: NONE

ADDRID: NONE

HOMEPH: NONE

☐ Parent object owns child object (cascade delete)

☐ Preserve MT_PHONE when parent is updated

☐ MT_PHONE required for operations on parent

An operation can be performed by a standard SQL statement or by a stored procedure. You can run a stored procedure to perform the operation or to do custom processing before or after processing.

To use a stored procedure, add it to the list and then configure it:

a. In **Choose parent table**, select the name of the table that is the parent of the table you are configuring. If you do not see the parent table in the list, the parent table has not yet been configured. Go back and configure the parent object before configuring the child objects.

b. Specify the cardinality of the relationship:
   - If the table has a single-cardinality relationship with the parent table, select **Single-cardinality**. In a single cardinality relationship, a parent can have only one child business object of this type. A single-cardinality relationship can be used with ownership to represent a true child, or without ownership to represent lookup tables or other peer objects in a database.
   - If the table is a multiple-cardinality relationship, do not select **Single cardinality**. In a multiple-cardinality relationship, a parent can have an array of child business objects of this type.

c. Build the foreign key relationship between the parent and child by specifying, for each child column, whether it is a foreign key in the parent table.
   - If the child column is not a foreign key, select `NONE`.
   - If a child column is a foreign key, select the column in the parent table that corresponds to the child column.

Note: The wizard can configure only a single parent table. If the child table has multiple parent tables, you must use the business object editor to configure the remaining parent tables after existing the wizard.

d. If the parent object owns the child object, then the child objects in the database are deleted when the parent is deleted. To indicate that this child is owned by its parent, select **Parent owns child object (cascade delete)**. Otherwise, clear this option to prevent child objects, such as lookup tables, from being deleted when their parent is deleted.

e. If you do not want child objects to be deleted as part of an Update operation, select **Preserve *child_table_name* when parent is updated**.

   When a parent table is updated, the adapter compares the child business objects present in the input with the child business objects returned from the database. By default, the adapter deletes any child objects returned from the database that are not present in the input business object.

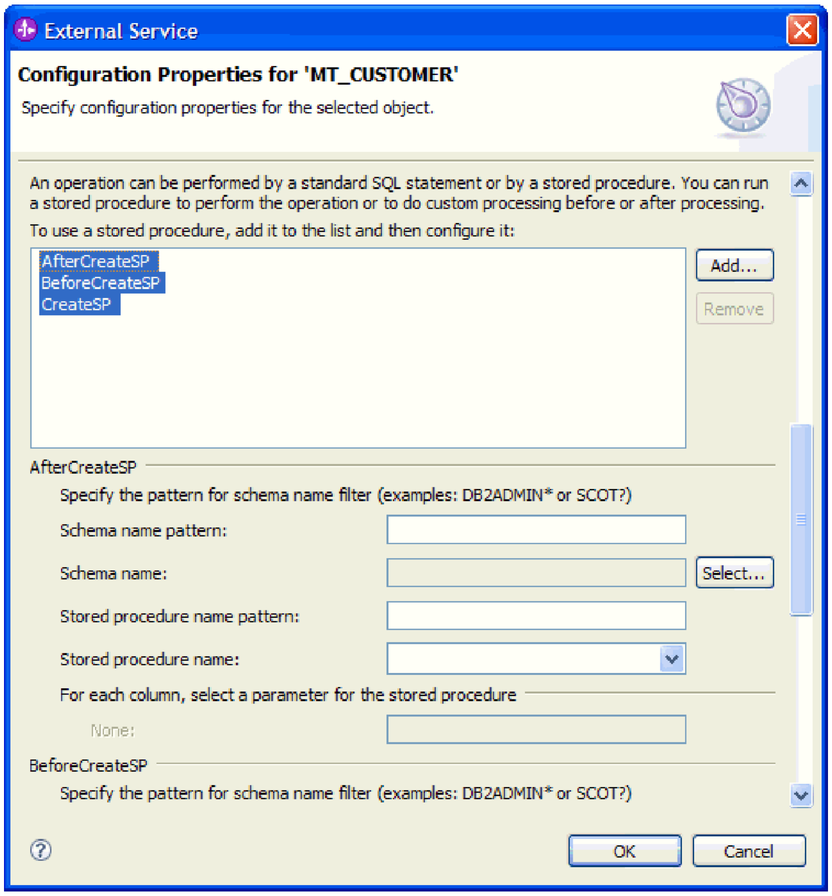
f. By default, you can perform operations on parent business objects without specifying the child business objects. If you want to requires that a parent business object must specify its child business objects when the parent is submitted for a change, select ***Child_table_name* required for operations on parent**.

5. An operation can be performed using either a standard SQL statement generated by the adapter or using stored procedures or stored functions from the database. If you want to use stored procedures or stored functions:

   a. Click **Add**.

   b. In the Add window, select the type of the stored procedure you want to run. For each operation, you can select a stored procedure that performs the operation, as well as stored procedures that run before or after the operation. For example, for the Create operation, you can specify any of CreateSP, BeforeCreateSP, and AfterCreateSP.

   c. Click **OK**. The Configuration Properties window now shows the stored procedure types you selected, and expands to display an area where you configure each one. It might be necessary to scroll down to see the new areas.


External Service

Configuration Properties for 'MT_CUSTOMER'

Specify configuration properties for the selected object.

An operation can be performed by a standard SQL statement or by a stored procedure. You can run a stored procedure to perform the operation or to do custom processing before or after processing.

To use a stored procedure, add it to the list and then configure it:

AfterCreateSP
BeforeCreateSP
CreateSP

AfterCreateSP

Specify the pattern for schema name filter (examples: DB2ADMIN* or SCOT?)

Schema name pattern:

Schema name:

Stored procedure name pattern:

Stored procedure name:

For each column, select a parameter for the stored procedure

None:

BeforeCreateSP

Specify the pattern for schema name filter (examples: DB2ADMIN* or SCOT?)

Important: In a hierarchical business object, if you want the stored procedure to be performed for each business object in the hierarchy, you must separately associate a stored procedure with the top-level business object and each child business object or array of business objects. If you associate a stored procedure with the top level business object but do not associate it with each child business object, then the top-level business object is processed with the stored procedure, but the child business objects are processed using the standard SQL query.

6. For each stored procedure type that you selected, specify the name of the stored procedure in the database and then configure the business object.
   a. Specify the name of the schema that contains the stored procedure. There are two ways to specify the name.
      - Filtering the database schemas to search for in the database.
        1) In **Schema name pattern**, type the name of the schema or a name pattern. Use the question mark (?) character to match a single character and the asterisk (*) to match multiple characters. The name is case-sensitive.
        2) In **Schema name**, select the name of the desired schema.
      - Selecting the schema from a list of all schemas discovered in the database.
        1) Leave the **Schema name pattern** field empty.
        2) Next to **Schema name**, click **Select** to open the Select window.
        3) In the Select window, select the name of the desired schema name. Optionally, type a case-sensitive pattern in **Value** to narrow down the

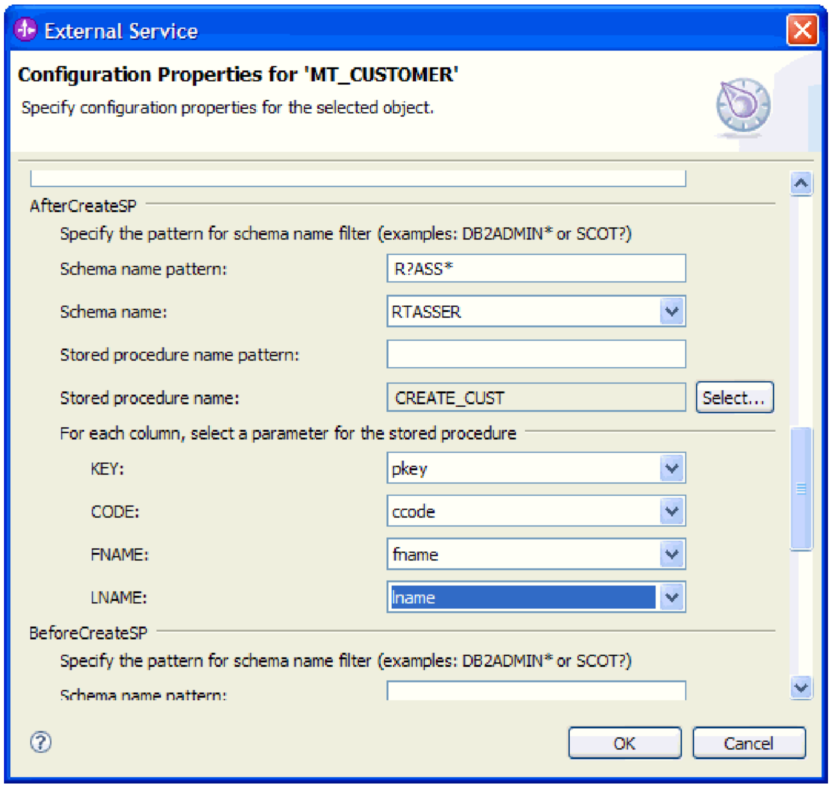
list. Use the question mark (?) character to match a single character and the asterisk (*) to match multiple characters.

4) When the desired schema is listed, select it and click **OK**.

b. Specify the name of the stored procedure or stored function. There are two ways to specify the name.
   - Filtering the stored procedures and stored functions to search for in the database.
     1) In **Stored procedure name pattern**, type the name of the stored procedure or stored function, or type a name pattern. Use the question mark (?) character to match a single character and the asterisk (*) to match multiple characters. The name is case-sensitive.
     2) In **Stored procedure name**, select the name of the desired procedure.
   - Selecting the name from a list of all stored procedures discovered in the database.
     1) Leave the **Stored procedure name pattern** field empty.
     2) Next to **Stored procedure name**, click **Select** to open the Select window.
     3) In the Select window, select the name of the desired stored procedure or stored function. Optionally, type a case-sensitive pattern in **Value** to narrow down the list. Use the question mark (?) character to match a single character and the asterisk (*) to match multiple characters.
     4) When the desired stored procedure or stored function is listed, select it and click **OK**.

   The Configuration Properties window expands to provide an area where you configure the stored procedure. The wizard automatically generates the list of parameters by examining the stored procedure in the database.

c. For each parameter in the stored procedure (on the left), select the table column (on the right) to pass to the stored procedure in that parameter. The following figure shows a portion of the window after a stored procedure has been configured.


External Service

Configuration Properties for 'MT_CUSTOMER'

Specify configuration properties for the selected object.

AfterCreateSP

Specify the pattern for schema name filter (examples: DB2ADMIN* or SCOT?)

Schema name pattern:	R?ASS*
Schema name:	RTASSER
Stored procedure name pattern:	
Stored procedure name:	CREATE_CUST

For each column, select a parameter for the stored procedure

KEY:	pkey
CODE:	ccode
FNAME:	fname
LNAME:	lname

BeforeCreateSP

Specify the pattern for schema name filter (examples: DB2ADMIN* or SCOT?)

Schema name pattern:

7. When all fields on the window are completed, click **OK** to save the configuration of the business object. The table, view, synonym, and nickname business objects you defined are now listed in the Object Discovery and Selection window.
8. To change the configuration of an object in the **Selected objects** area, select the object name and then click the (Edit) icon.
9. When you have selected and configured all business objects that you need, click **Next** to set global properties and configure wrapper business objects.

What to do next

Continue working in the Object Discovery and Selection window to select and configure other types of business objects.

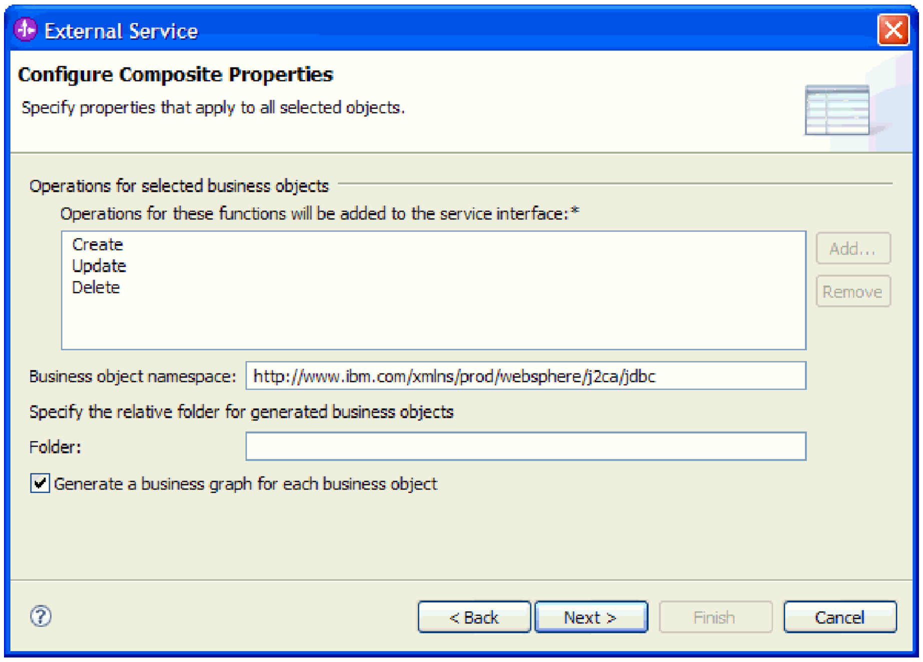
Setting global properties for operations

After you have selected database objects in the external service wizard, you need to specify properties that apply to all business objects.

Procedure

1. When the **Selected objects** list in the Object Discovery and Selection window contains all of the business objects you want to use in your application, except for wrapper business objects, click **Next**.
2. In the Configure Composite Properties window, review the list of operations. This list contains the operations that the adapter supports for the inbound services. To add to the list of operation includes operations for all business objects you selected on the previous window.

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


External Service

Configure Composite Properties

Specify properties that apply to all selected objects.

Operations for selected business objects

Operations for these functions will be added to the service interface:*

Create
Update
Delete

Business object namespace: http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc

Specify the relative folder for generated business objects

Folder:

[✓] Generate a business graph for each business object

3. To remove an operation that you do not need, select the operation name and click **Remove**. If you change your mind, click **Add** and restore a removed operation.
4. In **Business object namespace**, accept the default namespace or type the full name of another namespace.

   The namespace is prepended to the business object name to keep the business object schemas logically separated. For more information about this property, see "Business object namespace (BusinessObjectNameSpace)" on page 208.
5. Optionally, in **Folder**, type the relative path to the folder where the generated business objects are to be stored.
6. If you want a business graph to be created for each business object, click **Generate a business graph for each business object**. Business graphs are needed only when adding business objects to a module created with a version of WebSphere Integration Developer earlier than version 6.1.0

   Note: You must select this option if you are adding business objects to a module that was created with an earlier version of WebSphere Integration Developer. Otherwise, you must rewire your interface.
7. When you are finished, click **Next**.

Results

You have provided information that applies to all business objects in the module.

What to do next

Continue working in the wizard. The next step is to specify deployment information to use at run time and information for saving the service as a module.

Setting deployment properties and generating the service

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

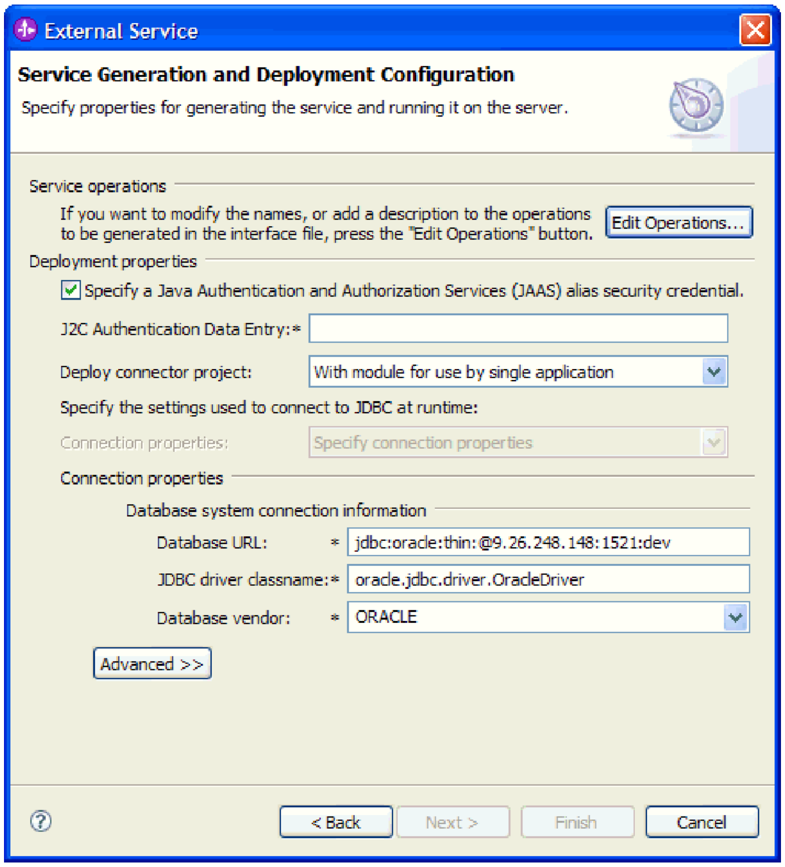
About this task

This task is performed using the Service Generation and Deployment Configuration and Service Location Properties windows of the external service wizard.

The connection properties in this task are initialized to the values that the wizard used to connect to the database. To configure the module to use other values, change the values here. For example, to use the IBM Toolkit for Java native driver at run time on i5/OS, set the driver information here.

Procedure

1. In the Service Generation and Deployment Configuration window, click **Edit Operations** to review the names of or add a description for the operations for the business objects you are creating.


External Service
Service Generation and Deployment Configuration
Specify properties for generating the service and running it on the server.
Service operations
If you want to modify the names, or add a description to the operations to be generated in the interface file, press the "Edit Operations" button.
Edit Operations...
Deployment properties
Specify a Java Authentication and Authorization Services (JAAS) alias security credential.
J2C Authentication Data Entry:*
Deploy connector project:
With module for use by single application
Specify the settings used to connect to JDBC at runtime:
Connection properties:
Specify connection properties
Connection properties
Database system connection information
Database URL: *
jdbc:oracle:thin:@9.26.248.148:1521:dev
JDBC driver classname:*
oracle.jdbc.driver.OracleDriver
Database vendor: *
ORACLE
Advanced >>
< Back
Next >
Finish
Cancel

2. Specify how you want the adapter to get the database user name and password at run time.
   - To use a J2C authentication alias, select **Specify a Java Authentication and Authorization Services (JAAS) alias security credential** and type the name of the alias in **J2C Authentication Data Entry**.

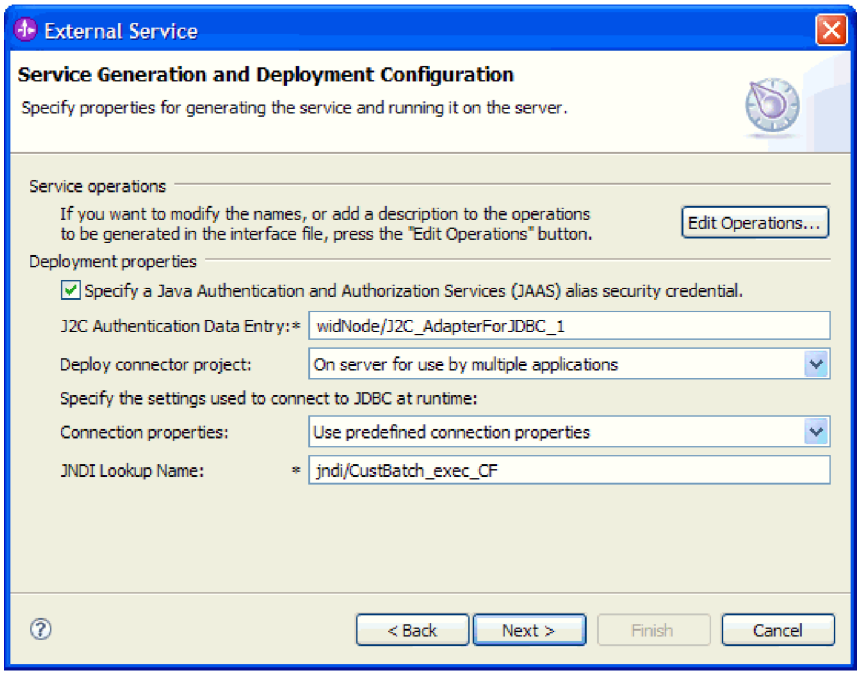
You can specify an existing authentication alias or create one at any time before deploying the module. The name is case-sensitive and includes the node name.

- To use the user name and password specified in an existing Java Naming and Directory Interface (JNDI) data source on the server:
  a. Clear **Specify a Java Authentication and Authorization Services (JAAS) alias security credential**.
  b. Click **Advanced**.
  c. Expand **Advanced connection configuration**.
  d. In **Datasource JNDI name**, type the name of an existing JNDI data source. For more information, see "Data source JNDI name (DataSourceJNDIName)" on page 209.
- To specify the database user name and password to be saved in the adapter properties:
  a. Clear **Specify a Java Authentication and Authorization Services (JAAS) alias security credential**.
  b. Click **Advanced**.
  c. Under **Database system connection information**, type the **User name** and **Password**. For more information, see "User name (UserName)" on page 220 and "Password (Password)" on page 216.

  Note: When you specify the password here, it is saved as clear text in an adapter property, which unauthorized users might be able to see.

3. In **Deploy connector project**, specify whether to include the adapter files in the module. Choose one of the following values:
   - **With module for use by single application**. With the adapter files embedded in the module, you can deploy the module to any application server. Use an embedded adapter when you have a single module using the adapter or if multiple modules need to run different versions of the adapter. Using an embedded adapter enables you to upgrade the adapter in a single module without the risk of destabilizing other modules by changing their adapter version.
   - **On server for use by multiple applications**. If you do not include the adapter files in a module, you must install them as a stand-alone adapter on each application server where you want to run the module. Use a stand-alone adapter when multiple modules can use the same version of the adapter and you want to administer the adapter in a central location. A stand-alone adapter can also reduce the resources required by running a single adapter instance for multiple modules.
4. If you specified **On server for use by multiple adapters** in the previous step, specify how you want to specify the connection properties.
   - If you manually created and configured a managed connection factory or activation specification on the server or if you have already deployed an application that connects to the same database using the same managed connection factory or activation specification properties, you can reuse the managed connection factory or activation specification by specifying the name of its Java Naming and Directory Interface (JNDI) data source:
     a. In **Connection properties**, select **Use predefined connection properties**.
     b. In **JNDI Lookup Name**, type the name of the JNDI data source for an existing managed connection factory or activation specification.

        The following figure shows typical settings for reusing a managed connection factory or activation specification for a stand-alone deployment of the adapter.


External Service

Service Generation and Deployment Configuration

Specify properties for generating the service and running it on the server.

Service operations

If you want to modify the names, or add a description to the operations to be generated in the interface file, press the "Edit Operations" button.

Deployment properties

[✓] Specify a Java Authentication and Authorization Services (JAAS) alias security credential.

J2C Authentication Data Entry:* widNode/J2C_AdapterForJDBC_1

Deploy connector project: On server for use by multiple applications

Specify the settings used to connect to JDBC at runtime:

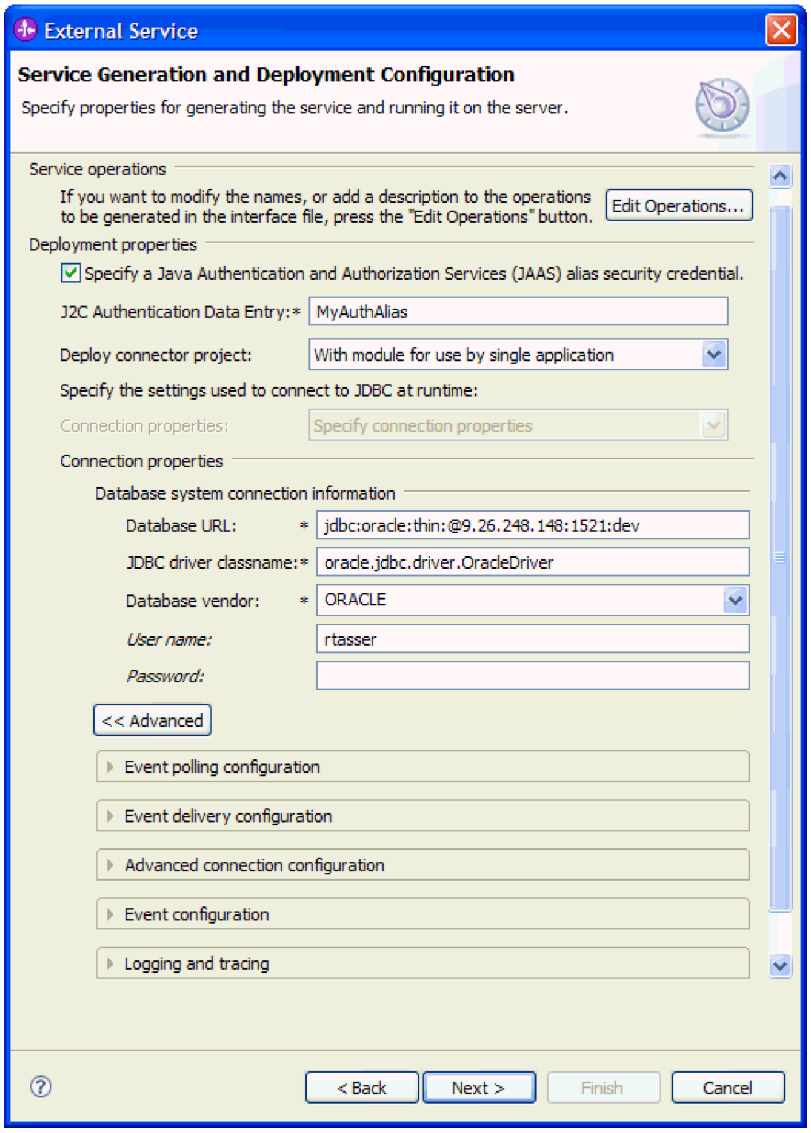
Connection properties: Use predefined connection properties

JNDI Lookup Name: * jndi/CustBatch_exec_CF

c. Click **Next** to complete this task.

- If this is the first application that connects to the database with a specific user name and password, or if you want to administer the user name and password separately from other applications, select **Specify connection properties**.

5. Review and, if necessary, change the values of the required connection properties. The fields are initialized with the connection information you specified when you started the wizard. You can change the values to specify a different user name and password at run time. You can also connect to an alternate database, although the schema names must be the same in both databases. The format of the connection properties are database-specific. For more information about the properties, see “Activation specification properties” on page 205.
6. Optionally, specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties. The following figure shows the advanced property sections on the Service Generation and Deployment Configuration window.



External Service

Service Generation and Deployment Configuration

Specify properties for generating the service and running it on the server.

Service operations

If you want to modify the names, or add a description to the operations to be generated in the interface file, press the "Edit Operations" button.

Deployment properties

☑ Specify a Java Authentication and Authorization Services (JAAS) alias security credential.

J2C Authentication Data Entry:*	MyAuthAlias
Deploy connector project:	With module for use by single application

Specify the settings used to connect to JDBC at runtime:

Connection properties:	Specify connection properties

Connection properties

Database system connection information

Database URL: *	jdbc:oracle:thin:@9.26.248.148:1521:dev
JDBC driver classname:*	oracle.jdbc.driver.OracleDriver
Database vendor: *	ORACLE
User name:	rtasser
Password:	

- **Event polling configuration**
  a. In **Interval between polling periods**, type the number of milliseconds that the adapter waits between polling periods. For more information, see "Interval between polling periods (PollPeriod)" on page 217.
  b. In **Maximum events in polling period**, type the number of events to deliver in each polling period. For more information, see "Maximum events in polling period (PollQuantity)" on page 217.
  c. In **Retry interval if connection fails**, type the number of milliseconds to wait before trying to connect after a connection failure during polling. For more information, see "Retry interval if connection fails (RetryInterval)" on page 218.
  d. In **Number of times to retry the system connection**, type the number of times to retry the connection before reporting a polling error. For more information, see "Number of times to retry the system connection (RetryLimit)" on page 218.
  e. If you want the adapter to stop if polling errors occur, select **Stop the adapter when an error is encountered while polling**. If you do not select this option, the adapter logs an exception but continues to run.

For more information, see "Stop the adapter when an error is encountered while polling (StopPollingOnError)" on page 219.

- **Event delivery configuration**
  a. In **Type of delivery**, select the delivery method. The methods are described in "Delivery type (DeliveryType)" on page 212.
  b. If you want to ensure that events are delivered only once and to only one export, select **Ensure once-only delivery**. This option might reduce performance but does not result in duplicate or missing event delivery. For more information, see "Ensure once-only event delivery (AssuredOnceDelivery)" on page 213.
  c. By default, the adapter processes all events that it finds when it polls. If you do not want it to process events that have timestamps later than the current time, select **Do not process events that have a timestamp in the future**. For more information, see "Do not process events that have a timestamp in the future (FilterFutureEvents)" on page 212.
  d. In **Event types to process**, type a comma-separated list of the business objects for which you want events delivered. Leave this field blank to receive events for all business object types.

     For example, if you want to receive events only when the Customer and Order tables, but not other tables, are changed in the database, set this field to `Customer,Order`.

     For more information, see "Event types to process (EventTypeFilter)" on page 214.
  e. Under **Number of connections for event delivery**, specify the minimum and maximum number of connections to use to deliver events. For more information, see "Minimum connections (MinimumConnections)" on page 216 and "Maximum connections (MaximumConnections)" on page 216.
- **Advanced connection configuration**
  a. **Datasource JNDI name** was discusses earlier in this topic.
  b. Set **Additional JDBC driver connection properties**. See more information about the property in "Additional JDBC driver connection properties (DriverConnectionProperties)" on page 207.
  c. Set **SQL query to verify the connection**. See more information about the property in "SQL query to verify the connection (PingQuery)" on page 217.
  d. In **Query timeout**, type the length of time, in seconds, that the adapter should wait for a response to a database query. See more information about the property in "Query timeout (QueryTimeOut)" on page 218.
  e. Set **Return business object even when the stored procedure result set is empty**. See more information about the property in "Return business object even when the stored procedure result set is empty (ReturnDummyBOForSP)" on page 219.
- **Event configuration**
  a. In **Event order by**, indicate the order in which events are retrieved and processed. This is a comma-separated list of column names from the event table, plus the keywords that control the sort order for each column. Use `asc` for ascending order and `desc` for descending order. For more information, see "Event order by (EventOrderBy)" on page 213.
  b. In **Event table name**, accept the default name of the table that contains the event store, or type a different table name. For more information, see "Event table name (EventTableName)" on page 214.

c. In **Stored procedure to run before poll**, the name of a stored procedure or stored function to run before the actual poll query is called. For more information, see “Stored procedure to run before polling (SPBeforePoll)” on page 220.

d. In **Stored procedure to run after poll**, specify the name of a stored procedure or stored function to run after each polling cycle. For more information, see “Stored procedure to run after polling (SPAfterPoll)” on page 219.

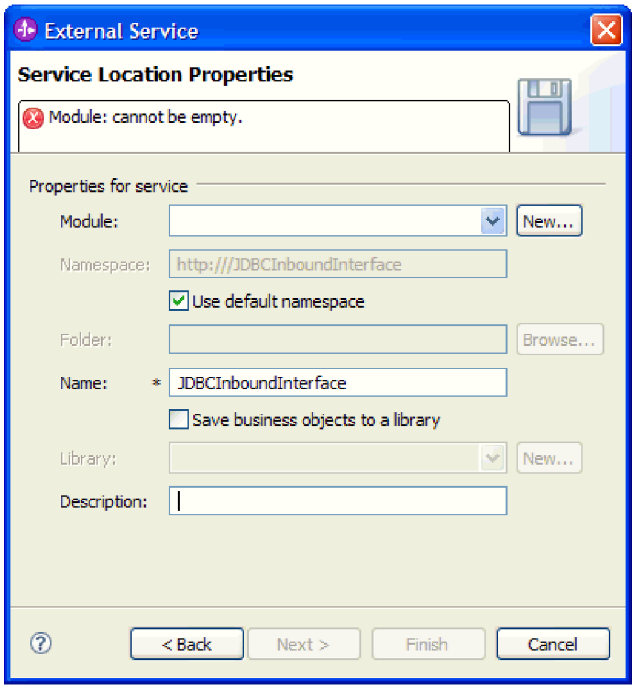
e. In **Event query type for processing events**, select the type of event processing you want to use:
   - To use the standard event processing provided by the adapter, select **Standard**.
   - To provide your own queries to customize event processing, select **User-Defined (Dynamic)**. If you select this option, complete the additional fields described in the following table.

Field	What to specify	For more information
Custom delete query	The name of the query, stored procedure, or stored function that is run after each event is processed to delete records that can be deleted after the event is delivered	“Custom delete query (CustomDeleteQuery)” on page 208
Custom event query	The name of the query, stored procedure, or stored function that performs the polling for events	“Custom event query (CustomEventQuery)” on page 208
Custom update query	The name of the query, stored procedure, or stored function that is run after each event is processed to prevent the event from being picked up for processing in a subsequent event cycle	“Custom update query (CustomUpdateQuery)” on page 209

- **Logging and tracing**

  a. In **Adapter ID to use for logging and tracing**, type a different adapter ID. See more information about the property in “Adapter ID to use for logging and tracing (AdapterID)” on page 185,

7. Click **Next**. The Service Location Properties window is displayed.


External Service

Service Location Properties

Module: cannot be empty.

Properties for service

Module:

Namespace: http:///JDBCInboundInterface

[✓] Use default namespace

Folder:

Name: * JDBCInboundInterface

[ ] Save business objects to a library

Library:

Description:

8. In the Service Location Properties window, specify the name of the module you want to create. This can be a new or existing module.
   - If the desired module name appears in the **Module** list, select its name.

     Important: If the module contains an interface or business object with the same name as any you are now configuring, the original interface or business object in the module will be replaced by the new version.
   - Otherwise, create a new module:
     a. Click **New**.
     b. In theIntegration Project window, select **Create a module project** and click **Next**.
     c. In the Module window, type a name for the module. For example, JDBCInboundModule.
     d. If you want the service description files (the .export and .wsdl files) to be located in the default folder in the module, leave **Use default location** selected. If you want to specify a different folder in the module, clear the option and then click **Browse** to specify a different folder in **Location**.
     e. If you want the module to be automatically opened in the assembly diagram in WebSphere Integration Developer when the wizard closes, select **Open module assembly diagram**. Otherwise, clear this option.
     f. Click **Finish** to create the new module.
9. Specify the namespace you want to use for your business objects.
   - If you want the business objects in the module to use the default namespace, leave **Use default namespace** selected.
   - To specify a different namespace, clear the option and type a different value in **Namespace**.
10. Optionally, specify the folder within the new module where the service description will be saved. In **Folder**, type the folder name or browse to an

existing folder. If you do not specify a folder name, the artifacts (the export, XSD, and WSDL files) are stored in the root folder of the module, that is, the folder with the module name.

11. In **Name**, accept the default name of the import or type a different name.
12. Optionally, if you want to save the business objects in a library where they can be used by other modules, select **Save business objects to a library** and specify the location of the library in **Library**.
13. Optionally, in **Description**, type a descriptive comment about the module.
14. When you are finished setting properties, click **Finish**.
15. If the Model Changed window is displayed, click **Yes**.

Results

The wizard exits. The module is created in the project and artifacts are generated.

What to do next

In some instances, you might need to use the assembly editor to complete the configuration. Then you can test or deploy your module.

Completing the configuration

In some situations, manual configuration steps are needed to complete the configuration of your business objects.

About this task

2 Perform this task when you need to customize the artifacts generated by the
2 wizard. You might do this in the following situations:
- 2 To set the CopyAttribute parameter for a column so that its value is set to the
2 same value as another column.
- To add or remove attributes from a business object. For example, you can simplify your business object design by removing the simple attribute corresponding to any database column that you do not need to reference.
- To configure additional parents for a table business object that has multiple parents. The wizard configures only one parent for a table business object.

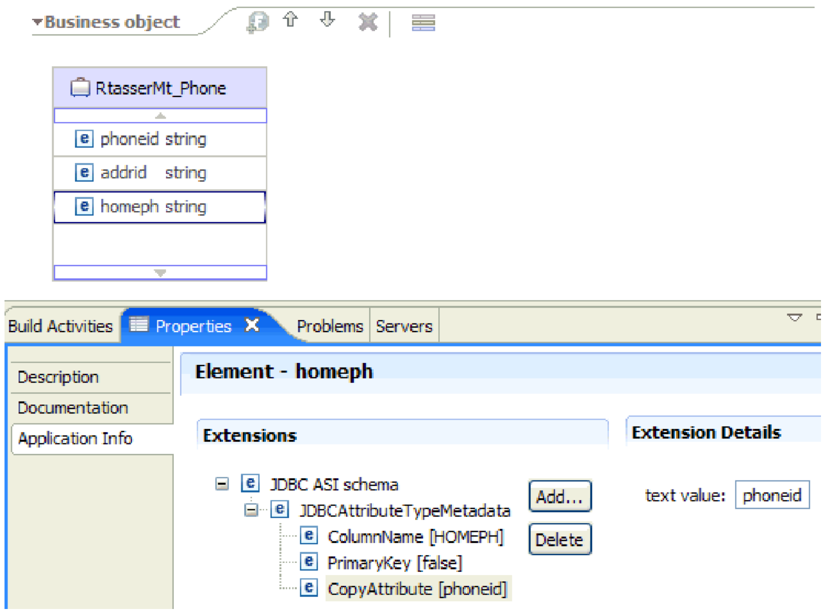
2 This topic provides detailed instructions for setting the CopyAttribute parameter to
2 a table business object. Other changes to business object structures can be
2 accomplished using similar techniques.

2 The CopyAttribute parameter is contained in the properties of the attribute for the
2 column that you want to populate with values and application-specific information
2 from another column. For example, if you want the `contact` column of a new row
2 in the table to contain the same value as the `email` column, set the CopyAttribute
2 parameter of the `contact` attribute to `email`. You use the assembly editor in
2 WebSphere Integration Developer to set the value.

Procedure

1. In the Business Integration perspective in WebSphere Integration Developer, expand the module name, expand **Data Type**, and then locate the table business object. The business object name is the name of the database schema plus the name of the database table. An optional namespace might be included at the beginning of the name.

2. Right-click the business object name and select **Open**. The assembly editor displays the business object, which has a field for each column.
3. 2 In the assembly editor, select the column you want to set to match another
2 column.
4. In the Properties view, select Application Info. If the Properties view is not visible, right-click the column name and click **Show in Properties**.
5. Expand **JDBC ASI schema**, and then expand **JDBCAttributeTypeMetadata**.
6. 2 Right-click **JDBCAttributeTypeMetadata** and then select **New** →
2 **jdbcasi:CopyAttribute**.
7. 2 Select the
8. 2 In the Extension Details area, set the text value to the name of the column that
2 contains the information to copy. The column can be in the current business
2 object or its parent business object. To copy from a column in the current
2 business object, set the value to the column name, for example, `phoneid`. To
2 copy from a column in the parent business object, prefix the column name with
2 two periods (..), for example, `..phone`. The following figure shows the assembly
2 editor showing the CopyAttribute property set to a column in the current table.
2



Element - homeph

Extensions

JDBC ASI schema
JDBCAttributeTypeMetadata
ColumnName [HOMEPH]
PrimaryKey [false]
CopyAttribute [phoneid]

Extension Details

text value: phoneid

2

2 Results

2 The business object is configured to use the CopyAttribute property to set the
2 business object attribute for a database column using information in another
2 column.

What to do next

You can now test and deploy the module.

Chapter 5. Changing interaction specification properties using the assembly editor

To change interaction specification properties for your adapter module after generating the service, use the assembly editor in WebSphere Integration Developer.

Before you begin

You must have used the external service wizard to generate a service for the adapter.

About this task

You might want to change interaction specification properties after you have generated a service for the adapter. Interaction specification properties, which are optional, are set at the method level, for a specific operation on a specific business object. The values you specify will appear as defaults in all parent business objects generated by the external service wizard. You can change these properties before you export the EAR file. You cannot change these properties after you deploy the application.

To change the interaction specification properties, use the following procedure.

Procedure

1. From the Business Integration perspective of WebSphere Integration Developer, expand the module name.
2. Expand **Assembly Diagram** and double-click the interface.
3. Click the interface in the assembly editor. (It shows the module properties if you don't do the extra click.)
4. Click the **Properties** tab. (You can also right-click the interface in the diagram and click **Show in Properties**.)
5. Under **Binding**, click **Method bindings**. The methods for the interface are displayed, one for each combination of business object and operation.
6. Select the method whose interaction specification property you want to change.
7. Change the property in the **Generic** tab. Repeat this step for each method whose interaction specification property you want to change.

Results

The interaction specification properties associated with your adapter module are changed.

What to do next

Deploy the module.

Chapter 6. Deploying the module

Deploy a module to place the files that make up your module and adapter into an operational environment for production or testing. In WebSphere Integration Developer, the integrated test environment features runtime support for WebSphere Process Server, or WebSphere Enterprise Service Bus, or both, depending on the test environment profiles that you selected during installation.

Deployment environments

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

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

Deploying the module for testing

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

Generating and wiring a target component for testing inbound processing

Before deploying to the test environment a module that includes an adapter for inbound processing, you must first generate and wire a target component. This target component serves as the *destination* to which the adapter sends events.

Before you begin

You must have generated an export module, using the external service wizard.

About this task

Generating and wiring a target component for inbound processing is required in a testing environment only. It is not necessary when deploying the adapter in a production environment.

The target component receives events. You *wire* the export to the target component (connecting the two components) using the assembly editor in WebSphere Integration Developer. The adapter uses the wire to pass event data (from the export to the target component).

Procedure

1. Create the target component
   a. From the Business Integration perspective of WebSphere Integration Developer, expand **Assembly Diagram** and double-click the export component. If you did not change the default value, the name of the export component is the name of your adapter + **InboundInterface**.

      An interface specifies the operations that can be called and the data that is passed, such as input arguments, returned values, and exceptions. The **InboundInterface** contains the operations required by the adapter to support inbound processing and is created when you run the external service wizard.
   b. Create a new component by expanding **Components**, selecting **Untyped Component**, and dragging the component to the Assembly Diagram.

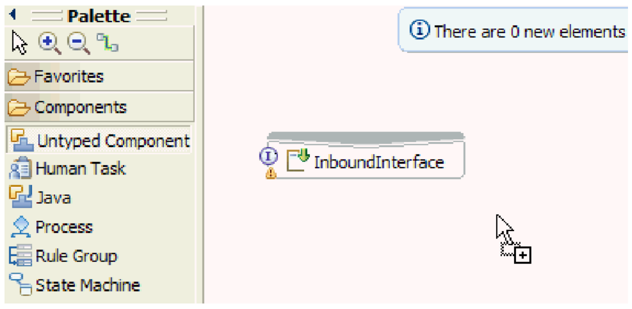


Figure 19. Adding a component to the Assembly Diagram

      The cursor changes to the placement icon.
   c. Click the component to have it displayed in the Assembly Diagram.
2. Wire the components.
   a. Click and drag the export component to the new component. This draws a wire from the export component to the new component, as shown in the following figure:

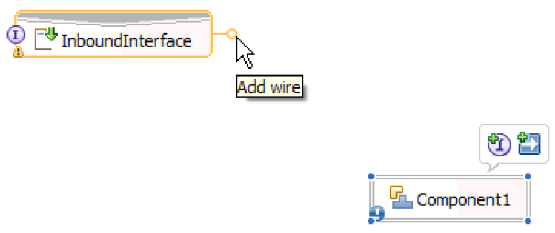


Figure 20. Selecting the wire icon

   b. Save the assembly diagram. Click **File** → **Save**
3. Generate an implementation for the new component.
   a. Right-click on the new component and select **Generate implementation** → **Java**.

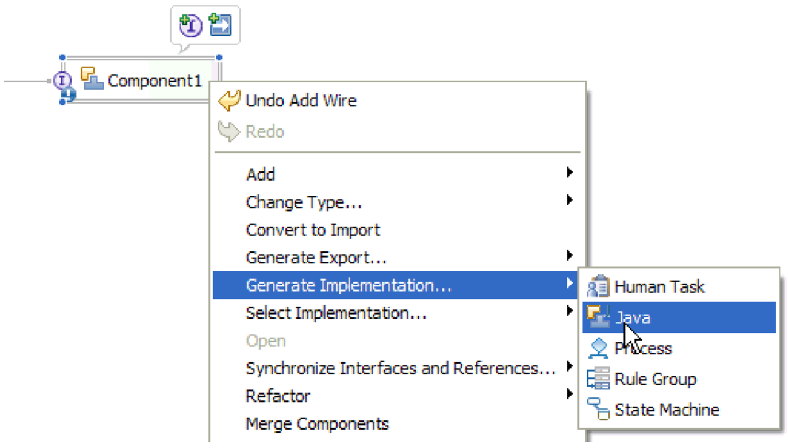


Figure 21. Generating a Java implementation

b. Select **(default package)** and click **OK**. This creates an endpoint for the inbound module.

   The Java implementation is displayed in a separate tab.

c. **Optional:** Add print statements to print the data object received at the endpoint for each of the endpoint methods.

d. Click **File** → **Save** to save the changes.

What to do next

Continue deploying the module for testing.

Preparing to test outbound operations

Before you can test your module's outbound processing with the WebSphere Integration Developer test client, you might need to modify some of your business objects.

About this task

This step is performed in the WebSphere Integration Developer test client. If it not already open, open it from the Business Integration perspective by right-clicking the name of your project and then clicking **Test** → **Test Module**.

- **Query business objects**

  If your query business object was created without a WHERE clause (for example, it was defined with a SELECT statement like `Select * from Customer`), unset the jdbcwhereclause attribute of the query business object before testing in the test client.

- **Table, view, and synonyms or nicknames business objects**

  Before testing the RetrieveAll operation, you need to unset any attribute whose value you are not setting as part of your test.

- **Query business objects**

  Before testing the RetrieveAll operation, you need to unset any attribute whose value you are not setting as part of your test.

Adding the module to the server

In WebSphere Integration Developer, you can add modules to one or more servers in the test environment.

Before you begin

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

About this task

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

Procedure

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

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

What to do next

Test the functionality of your module and the adapter.

Testing the module for outbound processing using the test client

Test the assembled module and adapter for outbound processing using the WebSphere Integration Developer integration test client.

Before you begin

You need to add the module to the server first.

About this task

Testing a module is generally performed on the interface operations of your components, which enables you to determine whether the components are correctly implemented and the references are correctly wired.

Procedure

1. Select the module you want to test, right-click on it, and select **Test → Test Module**.

2. For information on testing a module using the test client, see the *Testing modules and components* topic in the WebSphere Integration Developer information center.

What to do next

If you are satisfied with the results of testing your module and adapter, you can deploy the module and adapter to the production environment.

Deploying the module for production

Deploying a module created with the external service wizard to WebSphere Process Server or WebSphere Enterprise Service Bus in a production environment is a two-step process. First, you export the module in WebSphere Integration Developer as an enterprise archive (EAR) file. Second, you deploy the EAR file using the WebSphere Process Server administrative console.

Adding external software dependencies on the server

The adapter needs certain files installed on WebSphere Process Server or WebSphere Enterprise Service Bus server to be able to communicate with the database.

Before you begin

You do not need to perform this task if the database is installed on the same computer system as WebSphere Process Server or WebSphere Enterprise Service Bus. The files are already available to the adapter.

About this task

A stand-alone adapter needs the JDBC driver files and native system libraries from the database to be able to communicate with it. An embedded adapter needs the native system libraries.

Procedure

1. Obtain the JDBC driver-specific files and native libraries for your database software and operating system from your database administrator or from the database software Web site. The files that you need vary by database server. The following tables list the files needed for common database software. The following table by database server. The following table lists the JDBC driver files needed for common database software.

Table 12. JDBC driver files for common database software

Database software	Driver	JDBC driver files	Native System Libraries
IBM DB2 Universal Database for Linux, UNIX, and Windows	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cu.jar	None
IBM DB2 for z/OS	IBM DB2 Universal (Type 4)	db2jcc.jar db2jcc_license_cisuz.jar	None

Table 12. JDBC driver files for common database software (continued)

Database software	Driver	JDBC driver files	Native System Libraries
IBM DB2 for i5/OS	IBM Toolbox for Java remote driver	jt400.jar	None
	IBM DB2 Universal driver	db2jcc.jar db2jcc_license_cisuz.jar	None
	IBM Toolkit for Java native driver*	db2_classes.jar	None
Oracle	Thin driver	ojdbc14.jar	None
Microsoft SQL Server 2005	Microsoft SQL Server 2005 for JDBC	sqljdbc.jar	None
* You can use the IBM Toolkit for Java native driver to connect to the database at adapter run time, but you cannot use it to connect while running the wizard. You must use either the IBM Toolbox for Java remote driver or the IBM DB2 Universal driver during the discovery process. However, you can configure the module to use the native driver at run time. Do this on the Service Generation and Deployment Configuration window.			

2. For a stand-alone adapter, copy the JDBC driver files to WebSphere Process Server or WebSphere Enterprise Service Bus.
3. If the module is configured to use a type 2 driver, copy the native system libraries to the server. This step is needed for both stand-alone and embedded adapters.

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

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

Before you begin

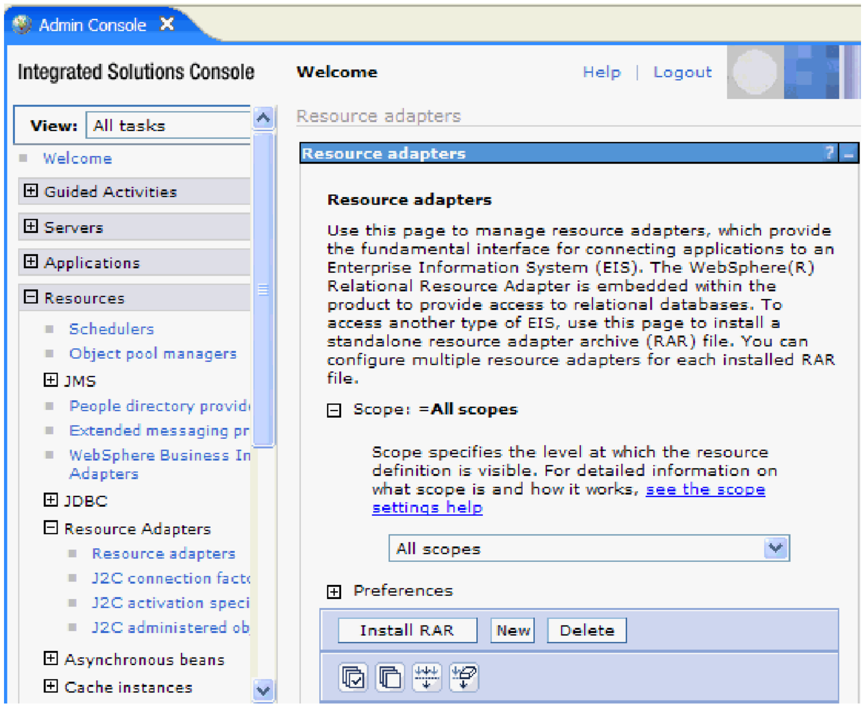
You must have set **Deploy connector project** to **On server for use by multiple adapters** in the Service Generation and Deployment Configuration window of the external service wizard.

About this task

Installing the adapter in the form of a RAR file results in the adapter being available to all J2EE application components running in the server runtime.

Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. From the Resource adapters page, click **Install RAR**.



Resource adapters

Use this page to manage resource adapters, which provide the fundamental interface for connecting applications to an Enterprise Information System (EIS). The WebSphere(R) Relational Resource Adapter is embedded within the product to provide access to relational databases. To access another type of EIS, use this page to install a standalone resource adapter archive (RAR) file. You can configure multiple resource adapters for each installed RAR file.

Scope: =**All scopes**

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, see the scope settings help

All scopes

Figure 22. The Install RAR button on the Resource adapters page

4. From the Install RAR file page, click **Browse** and navigate to the RAR file for your adapter.

   The RAR files are typically installed in the following path: *WID_installation_directory*/ResourceAdapters/*adapter_name*/deploy/*adapter*.rar
5. Click **Next**.
6. From the Resource adapters page, optionally change the name of the adapter and add a description.
7. Click **OK**.
8. Click **Save** in the **Messages** box at the top of the page.

What to do next

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

Exporting the module as an EAR file

Using WebSphere Integration Developer, export your module as an EAR file. By creating an EAR file, you capture all of the contents of your module in a format that can be easily deployed to WebSphere Process Server or WebSphere Enterprise Service Bus.

Before you begin

Before you can export a module as an EAR file, you must have created a module to communicate with your service. The module should be displayed in the WebSphere Integration Developer Business Integration perspective.

About this task

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

Procedure

1. Right-click the module and select **Export**.
2. In the Select window, expand **J2EE**.
3. Select **EAR file** and click **Next**.

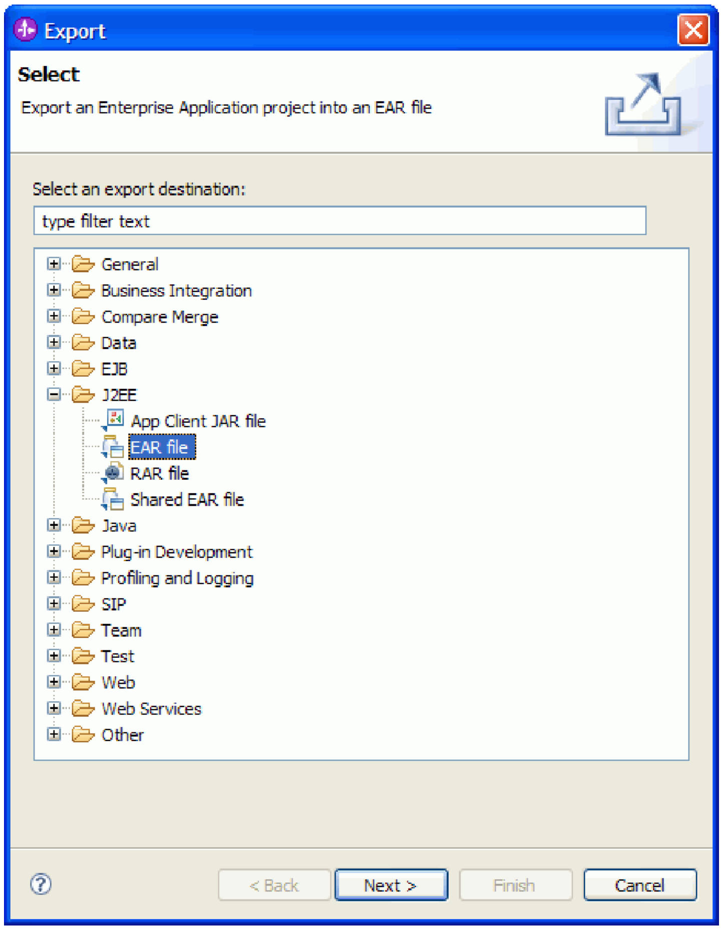


Figure 23. Selecting **EAR file** *from the Select window*

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

Results

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

What to do next

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

Installing the EAR file

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

Before you begin

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

About this task

To install the EAR file, perform the following procedure. For more information on clustering adapter module applications, see the http://www.ibm.com/software/webservers/appserv/was/library/.

Procedure

1. Open the WebSphere Process Server administrative console by right-clicking your server instance and selecting **Run administrative console**.
2. In the administrative console window, click **Applications** → **Install New Applications**.

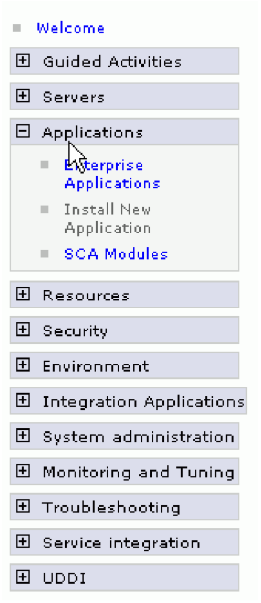


Figure 24. Preparing for the application installation window

3. Click **Browse** to locate your EAR file and click **Next**. The EAR file name is the name of the module followed by "App."
4. Optional: If you are deploying to a clustered environment, complete the following steps.
   a. On the **Step 2: Mapping modules to servers** window, select the module.

   b. Select the name of the server cluster.
   c. Click **Apply**.
5. Click **Next** to open the Summary. Verify that all settings are correct and click **Finish**.
6. Optional: If you are using an authentication alias, complete the following steps:
   a. Expand **Security** and select **Business Integration Authentication Aliases**.
   b. Select the authentication alias that you want to configure. You must have administrator or operator authority to make changes to authentication alias configurations.
   c. Optional: If it is not already filled in, type the **User name**.
   d. If it is not already filled in, type the **Password**.
   e. If it is not already filled in, type the password again in the **Confirm Password** field.
   f. Click **OK**.

Results

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

What to do next

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

Chapter 7. Administering the adapter module

When you are running the adapter in a stand-alone deployment, use the administrative console of the server to start, stop, monitor, and troubleshoot the adapter module. In an application that uses an embedded adapter, the adapter module starts or stops when the application is started or stopped.

Changing configuration properties for embedded adapters

To change configuration properties after you deploy the adapter as part of a module, you use the administrative console of the runtime environment. You can update resource adapter properties (used for general adapter operation), managed connection factory properties (used for outbound processing), and activation specification properties (used for inbound processing).

Setting resource adapter properties for embedded adapters

To set resource adapter properties for your adapter after it has been deployed as part of a module, use the administrative console. You select the name of the property you want to configure and then change or set the value.

Before you begin

Your adapter module must be deployed on WebSphere Process Server or WebSphere Enterprise Service Bus.

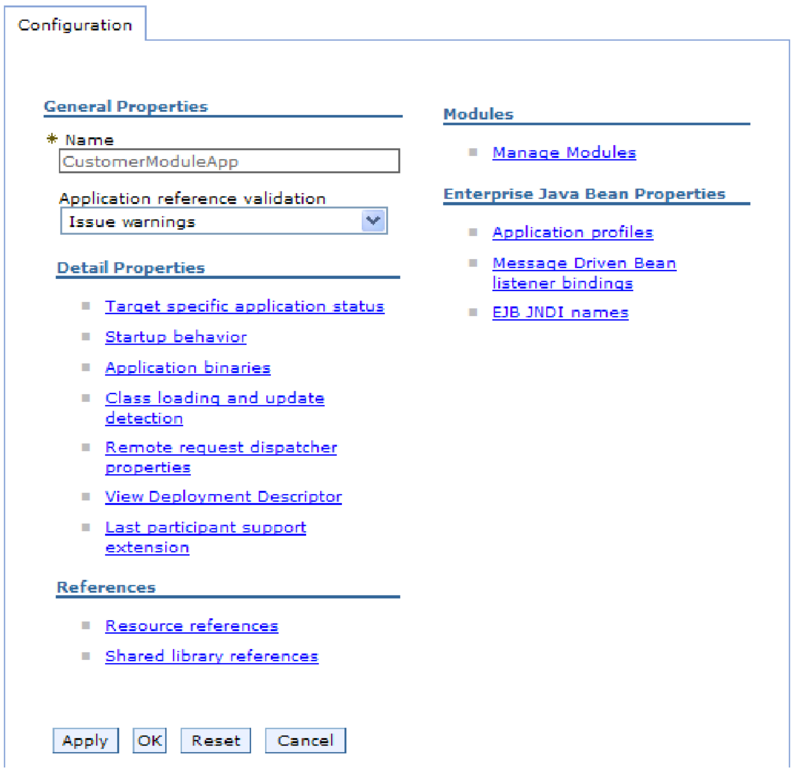
About this task

Custom properties are default configuration properties shared by all WebSphere adapters.

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Under **Applications**, select **Enterprise Applications**.
3. From the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change.
4. Under **Modules**, click **Manage Modules**.


Configuration
General Properties
* Name
Application reference validation
Detail Properties
Target specific application status
Startup behavior
Application binaries
Class loading and update detection
Remote request dispatcher properties
View Deployment Descriptor
Last participant support extension
References
Resource references
Shared library references
Modules
Manage Modules
Enterprise Java Bean Properties
Application profiles
Message Driven Bean listener bindings
EJB JNDI names
Apply
OK
Reset
Cancel

Name: CustomerModuleApp

Application reference validation: Issue warnings

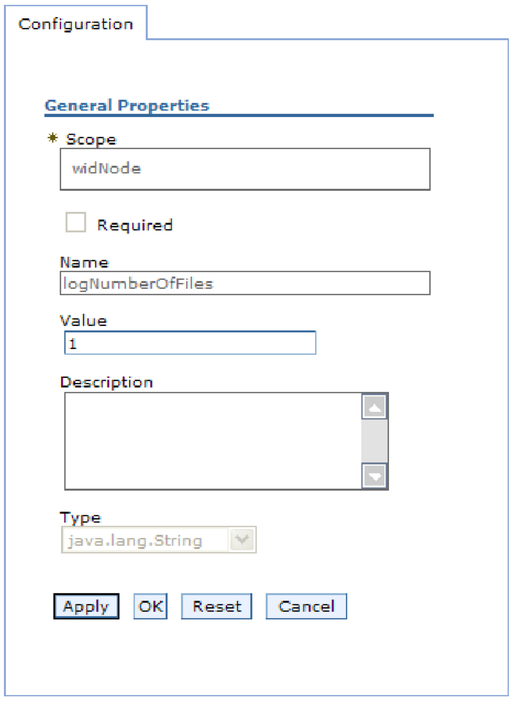
Figure 25. The Manage Modules selection in the Configuration tab

5. Click **IBM WebSphere Adapter for JDBC**.
6. From the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **Custom properties**.
8. For each property you want to change, perform the following steps.

   Note: See "Resource adapter properties" on page 184 for more information about these properties.

   a. Click the name of the property.
   b. Change the contents of the **Value** field or type a value, if the field is empty.

      For example, if you click **logNumberOfFiles**, you see the following page:


Configuration

General Properties

* Scope: widNode

Required: [ ]

Name: logNumberOfFiles

Value: 1

Description:

Type: java.lang.String

Figure 26. The Configuration tab for the logNumberOfFiles property

You can change the number in the **Value** field and add a description of the property.

c. Click **OK**.

9. Click the **Save** link in the **Messages** box at the top of the window.

Results

The resource adapter properties associated with your adapter module are changed.

Setting managed (J2C) connection factory properties for embedded adapters

To set managed connection factory properties for your adapter after it has been deployed as part of a module, use the administrative console. You select the name of the property you want to configure and then change or set the value.

Before you begin

Your adapter module must be deployed on WebSphere Process Server or WebSphere Enterprise Service Bus.

About this task

You use managed connection factory properties to configure the target database instance.

Note: In the administrative console, the properties are referred to as "J2C connection factory properties."

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Under **Applications**, select **Enterprise Applications**.
3. From the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change.
4. Under **Modules**, click **Manage Modules**.

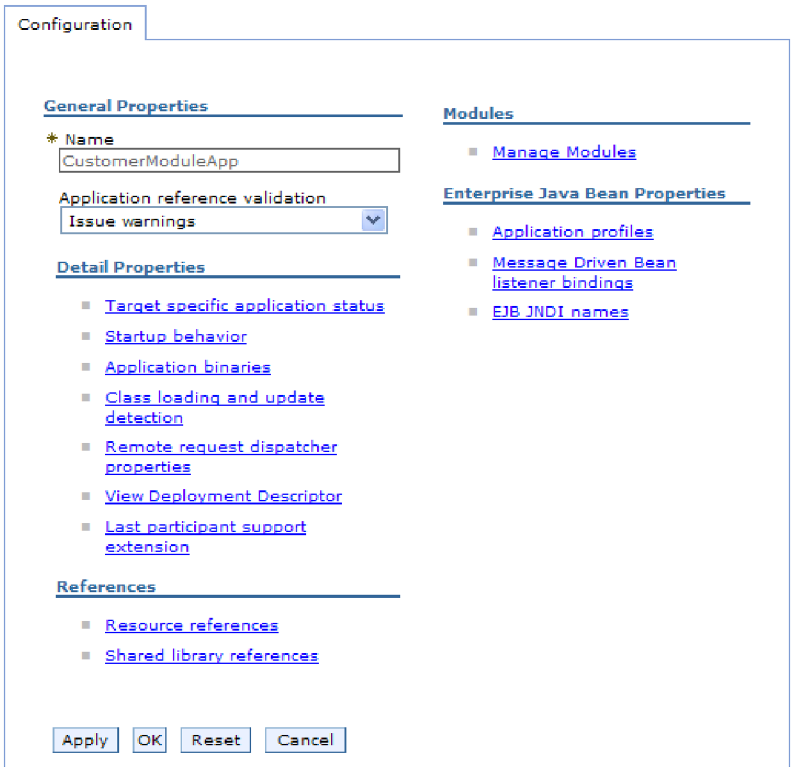

Configuration
General Properties
* Name
CustomerModuleApp
Application reference validation
Issue warnings
Detail Properties
Target specific application status
Startup behavior
Application binaries
Class loading and update detection
Remote request dispatcher properties
View Deployment Descriptor
Last participant support extension
References
Resource references
Shared library references
Modules
Manage Modules
Enterprise Java Bean Properties
Application profiles
Message Driven Bean listener bindings
EJB JNDI names
Apply
OK
Reset
Cancel

Figure 27. The Manage Modules selection in the Configuration tab

5. Click **IBM WebSphere Adapter for JDBC**.
6. From the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **J2C connection factories**.
8. Click the name of the connection factory associated with your adapter module.
9. From the **Additional Properties** list, click **Custom properties**.

   Custom properties are those J2C connection factory properties that are unique to Adapter for JDBC. Connection pool and advanced connection factory properties are properties you configure if you are developing your own adapter.
10. For each property you want to change, perform the following steps.

    Note: See "Managed connection factory properties" on page 187 for more information about these properties.

    a. Click the name of the property.

b. Change the contents of the **Value** field or type a value, if the field is empty.
c. Click **OK**.

11. Click the **Save** link in the **Messages** box at the top of the window.

Results

The managed connection factory properties associated with your adapter module are changed.

Setting activation specification properties for embedded adapters

To set activation specification properties for your adapter after it has been deployed as part of a module, use the administrative console. You select the name of the message endpoint property you want to configure, and then change or set the value.

Before you begin

Your adapter module must be deployed on WebSphere Process Server or WebSphere Enterprise Service Bus.

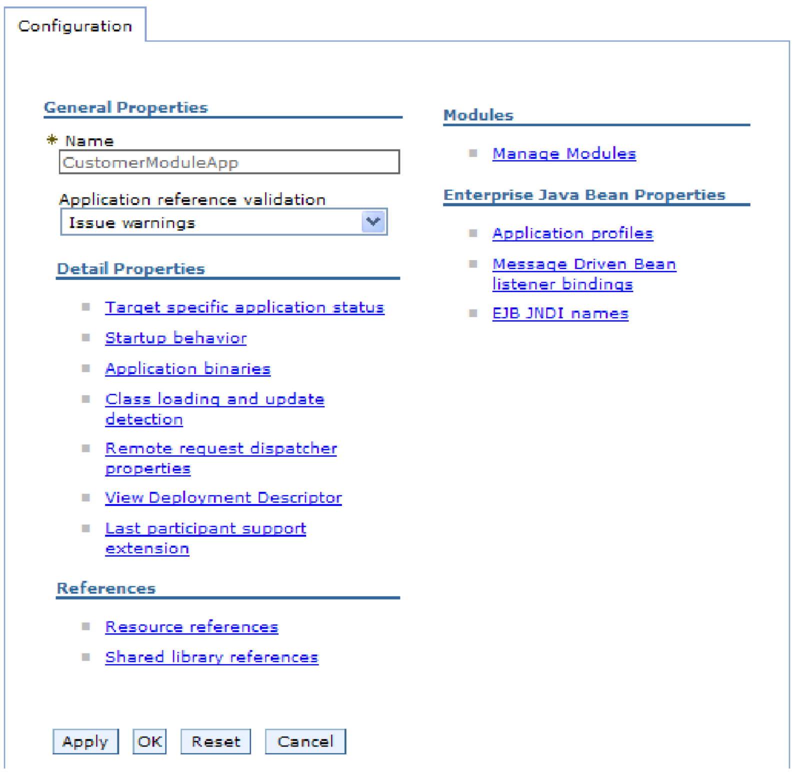
About this task

You use activation specification properties to configure the endpoint for inbound processing.

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Under **Applications**, select **Enterprise Applications**.
3. From the **Enterprise Applications** list, click the name of the adapter module whose properties you want to change.
4. Under **Modules**, click **Manage Modules**.


Configuration

General Properties

* Name

CustomerModuleApp

Application reference validation

Issue warnings

Detail Properties

- Target specific application status
- Startup behavior
- Application binaries
- Class loading and update detection
- Remote request dispatcher properties
- View Deployment Descriptor
- Last participant support extension

References

- Resource references
- Shared library references

Modules

- Manage Modules

Enterprise Java Bean Properties

- Application profiles
- Message Driven Bean listener bindings
- EJB JNDI names

Figure 28. The Manage Modules selection in the Configuration tab

5. Click **IBM WebSphere Adapter for JDBC**.
6. From the **Additional Properties** list, click **Resource Adapter**.
7. On the next page, from the **Additional Properties** list, click **J2C activation specifications**.
8. Click the name of the activation specification associated with the adapter module.
9. From the **Additional Properties** list, click **J2C activation specification custom properties**.
10. For each property you want to change, perform the following steps.

    Note: See "Activation specification properties" on page 205 for more information about these properties.

    a. Click the name of the property.
    b. Change the contents of the **Value** field or type a value, if the field is empty.
    c. Click **OK**.
11. Click the **Save** link in the **Messages** box at the top of the window.

Results

The activation specification properties associated with your adapter module are changed.

Changing configuration properties for stand-alone adapters

To set configuration properties after you install a stand-alone adapter, you use the administrative console of the runtime environment. You provide general information about the adapter and then set resource adapter properties (which are used for general adapter operation). If the adapter will be used for outbound operations, you create a connection factory and then set properties for it. If the adapter will be used for inbound operations, you create an activation specification and then set properties for it.

Setting resource adapter properties for stand-alone adapters

To set resource adapter properties for your stand-alone adapter after it has been installed on WebSphere Process Server or WebSphere Enterprise Service Bus, use the administrative console. You select the name of the property you want to configure and then change or set the value.

Before you begin

Your adapter must be installed on WebSphere Process Server or WebSphere Enterprise Service Bus.

About this task

Custom properties are default configuration properties shared by all WebSphere adapters.

To configure properties using the administrative console, use the following procedure.

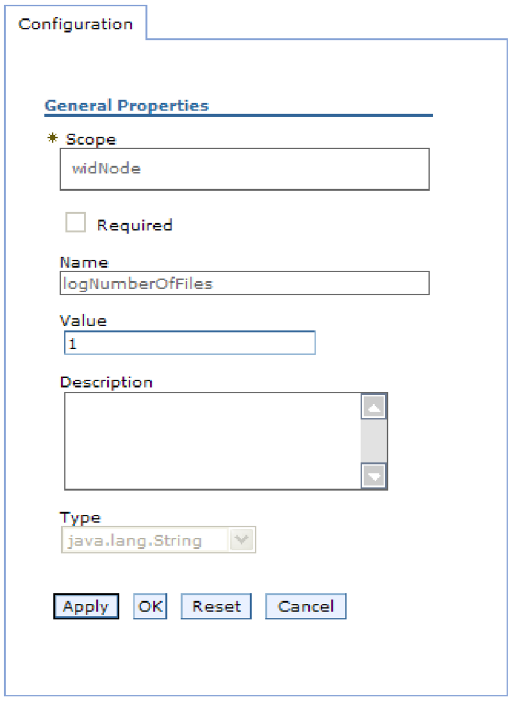
Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. From the Resource adapters page, click **IBM WebSphere Adapter for JDBC**.
4. From the **Additional Properties** list, click **Custom properties**.
5. For each property you want to change, perform the following steps.

   Note: See "Resource adapter properties" on page 184 for more information about these properties.

   a. Click the name of the property.
   b. Change the contents of the **Value** field or type a value, if the field is empty.

      For example, if you click **logNumberOfFiles**, you see the following page:


Configuration

General Properties

* Scope: widNode

Required: [ ]

Name: logNumberOfFiles

Value: 1

Description:

Type: java.lang.String

Figure 29. The Configuration tab for the logNumberOfFiles property

You can change the number in the **Value** field and add a description of the property.

c. Click **OK**.

6. Click **Save** in the **Messages** box at the top of the page.

Results

The resource adapter properties associated with your adapter are changed.

Setting managed (J2C) connection factory properties for stand-alone adapters

To set managed connection factory properties for your stand-alone adapter after it has been installed on WebSphere Process Server or WebSphere Enterprise Service Bus, use the administrative console. You select the name of the property you want to configure and then change or set the value.

Before you begin

Your adapter must be installed on WebSphere Process Server or WebSphere Enterprise Service Bus.

About this task

You use managed connection factory properties to configure the target database instance.

Note: In the administrative console, the properties are referred to as "J2C connection factory properties."

To configure properties using the administrative console, use the following procedure.

Procedure

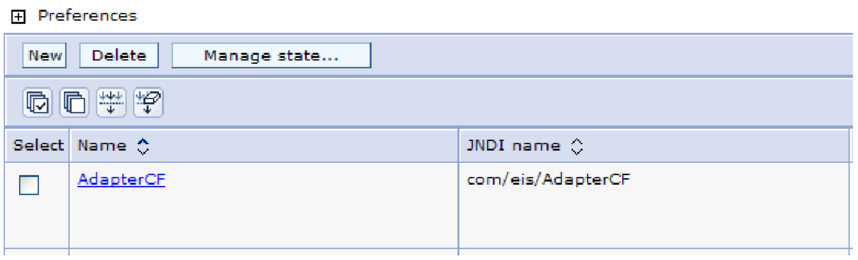
1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. From the Resource adapters page, click **IBM WebSphere Adapter for JDBC**.
4. From the **Additional Properties** list, click **J2C connection factories**.
5. If you are going to use an existing connection factory, skip ahead to step 6.

   Note: If you selected **Use predefined connection properties** when you used the external service wizard to configure the adapter module, you do not need to create a connection factory.

   If you are creating a connection factory, perform the following steps:

   a. Click **New**.
   b. In the **General Properties** section of the **Configuration** tab, type a name for the connection factory. For example, you could type `AdapterCF`.
   c. Type a value for **JNDI name**. For example, you could type `com/eis/AdapterCF`.
   d. Select an authentication alias from the **Component-managed authentication alias** list.
   e. Click **OK**.
   f. Click **Save** in the **Messages** box at the top of the page.

      The newly created connection factory is displayed.



Select	Name	JNDI name
☐	AdapterCF	com/eis/AdapterCF

Figure 30. The list of connection factories

6. From the list of connection factories, click the one you want to use.
7. From the **Additional Properties** list, click **Custom properties**.

   Custom properties are those J2C connection factory properties that are unique to Adapter for JDBC. Connection pool and advanced connection factory properties are properties you configure if you are developing your own adapter.
8. For each property you want to change, perform the following steps.

   Note: See “Managed connection factory properties” on page 187 for more information about these properties.

   a. Click the name of the property.
   b. Change the contents of the **Value** field or type a value, if the field is empty.
   c. Click **OK**.
9. After you have finished setting properties, click **Apply**.

10. Click **Save** in the **Messages** box at the top of the window.

Results

The managed connection factory properties associated with your adapter are set.

Setting activation specification properties for stand-alone adapters

To set activation specification properties for your stand-alone adapter after it has been installed on WebSphere Process Server or WebSphere Enterprise Service Bus, use the administrative console. You select the name of the message endpoint property you want to configure, and then change or set the value.

Before you begin

Your adapter must be installed on WebSphere Process Server or WebSphere Enterprise Service Bus.

About this task

You use activation specification properties to configure the endpoint for inbound processing.

To configure properties using the administrative console, use the following procedure.

Procedure

1. Start the administrative console.
2. Click **Resources** → **Resource Adapters** → **Resource adapters**.
3. From the Resource adapters page, click **IBM WebSphere Adapter for JDBC**.
4. From the **Additional Properties** list, click **J2C activation specifications.**.
5. If you are going to use an existing activation specification, skip ahead to step 6.

   Note: If you selected **Use predefined connection properties** when you used the external service wizard to configure the adapter module, you do not need to create an activation specification.

   If you are creating an activation specification, perform the following steps:
   a. Click **New**.
   b. In the **General Properties** section of the **Configuration** tab, type a name for the activation specification. For example, you could type `AdapterAS`.
   c. Type a value for **JNDI name**. For example, you could type `com/eis/AdapterAS`.
   d. Select an authentication alias from the **Authentication alias** list.
   e. Select a message listener type.
   f. Click **OK**.
   g. Click **Save** in the **Messages** box at the top of the page.

      The newly created activation specification is displayed.
6. From the list of activation specifications, click the one you want to use.
7. From the Additional Properties list, click **J2C activation specification custom properties**.

8. For each property you want to set, perform the following steps.

   Note: See "Activation specification properties" on page 205 for more information about these properties.

   a. Click the name of the property.
   b. Change the contents of the **Value** field or type a value, if the field is empty.
   c. Click **OK**.
9. After you have finished setting properties, click **Apply**.
10. Click **Save** in the **Messages** box at the top of the page.

Results

The activation specification properties associated with your adapter are set.

Starting the application that uses the adapter

Use the administrative console of the server to start an application that uses the adapter. By default, the application starts automatically when the server starts.

About this task

Use this procedure to start the application, whether it is using an embedded or a stand-alone adapter. For an application that uses an embedded adapter, the adapter starts when the application starts. For an application that uses a stand-alone adapter, the adapter starts when the application server starts.

Procedure

1. On the administrative console, click **Applications** → **Enterprise Applications**.

   Note: The administrative console is labeled "Integrated Solutions Console".
2. Select the check box of the application that you want to start. The application name is the name of the EAR file you installed, without the .EAR file extension.
3. Click **Start**.

Results

The status of the application changes to Started, and a message stating that the application has started displays at the top of the administrative console.

Stopping the application that uses the adapter

Use the administrative console of the server to stop an application that uses the adapter. By default, the application stops automatically when the server stops.

About this task

Use this procedure to stop the application, whether it is using an embedded or a stand-alone adapter. For an application with an embedded adapter, the adapter stops when the application stops. For an application that uses a stand-alone adapter, the adapter stops when the application server stops.

Procedure

1. On the administrative console, click **Applications** → **Enterprise Applications**.

Note: The administrative console is labeled "Integrated Solutions Console".

2. Select the check box of the application that you want to stop. The application name is the name of the EAR file you installed, without the .EAR file extension.
3. Click **Stop**.

Results

The status of the application changes to Stopped, and a message stating that the application has stopped displays at the top of the administrative console.

Monitoring performance using Performance Monitoring Infrastructure

Performance Monitoring Infrastructure (PMI) is a feature of the administrative console that allows you to dynamically monitor the performance of components in the production environment, including the adapter for JDBC. PMI collects adapter performance data, such as average response time and total number of requests, from various components in the server and organizes the data into a tree structure. You can view the data through the Tivoli® Performance Viewer, a graphical monitoring tool that is integrated with the administrative console in WebSphere Process Server.

About this task

You can monitor the performance of your adapter by having PMI collect data at the following points:

- At outbound processing to monitor outbound requests
- At inbound event retrieval to monitor the retrieval of an event from the event table
- At inbound event delivery to monitor the delivery of an event to the endpoint or endpoints

Before you can enable and configure PMI for your adapter, you must first set the level of tracing detail and run some events from which to gather performance data.

To learn more about how PMI can help you monitor and improve the overall performance of your adapter environment, search for PMI on the WebSphere Application Server web site: http://www.ibm.com/software/webservers/appserv/was/library/.

Configuring Performance Monitoring Infrastructure

You can configure Performance Monitoring Infrastructure (PMI) to gather adapter performance data, such as average response time and total number of requests. After you configure PMI for your adapter, you can monitor the adapter performance using Tivoli Performance viewer.

Before you begin

Before you can configure PMI for your adapter, you must first set the level of tracing detail and run some events from which to gather performance data.

1. To enable tracing and to receive event data, the trace level must be set to either fine, finer, finest, or all. After *=info, add a colon and a string, for example:

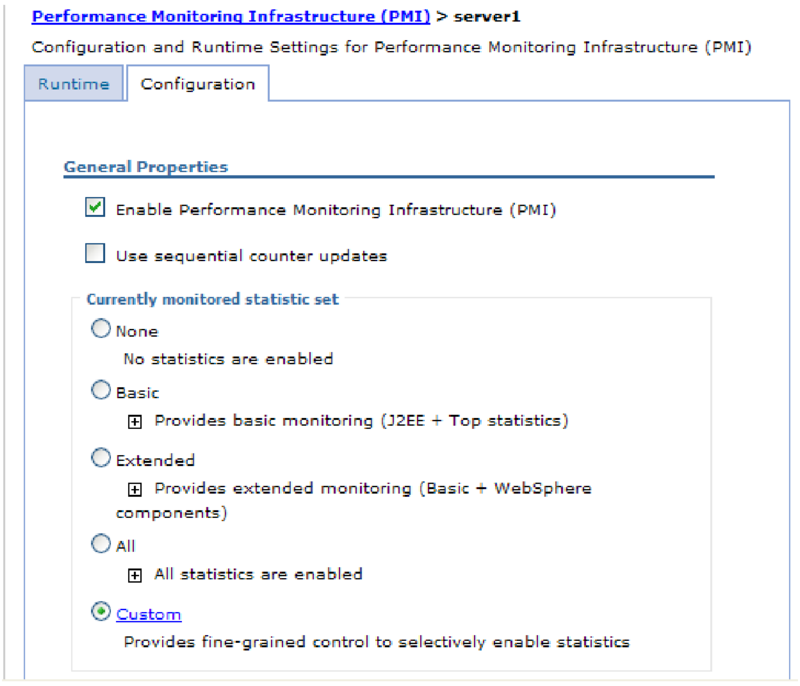
```
*=info: WBILocationMonitor.CEI.ResourceAdapter.
*=finest: WBILocationMonitor.LOG.ResourceAdapter.*=finest:
```

For instructions on setting the trace level, refer to “Enabling tracing with the Common Event Infrastructure (CEI)” on page 145.

2. Generate at least one outbound request or inbound event to produce performance data that you can configure.

Procedure

1. Enable PMI for your adapter.
   a. In the administrative console, expand **Monitoring and Tuning**, and then select **Performance Monitoring Infrastructure (PMI)**.
   b. From the list of servers, click the name of your server.
   c. Select the Configuration tab, then select the **Enable Performance Monitoring (PMI)** check box.
   d. Select **Custom** to selectively enable or disable statistics.



Performance Monitoring Infrastructure (PMI) > server1

Configuration and Runtime Settings for Performance Monitoring Infrastructure (PMI)

☑ Enable Performance Monitoring Infrastructure (PMI)

☐ Use sequential counter updates

Currently monitored statistic set

○ None
No statistics are enabled

○ Basic
⊞ Provides basic monitoring (J2EE + Top statistics)

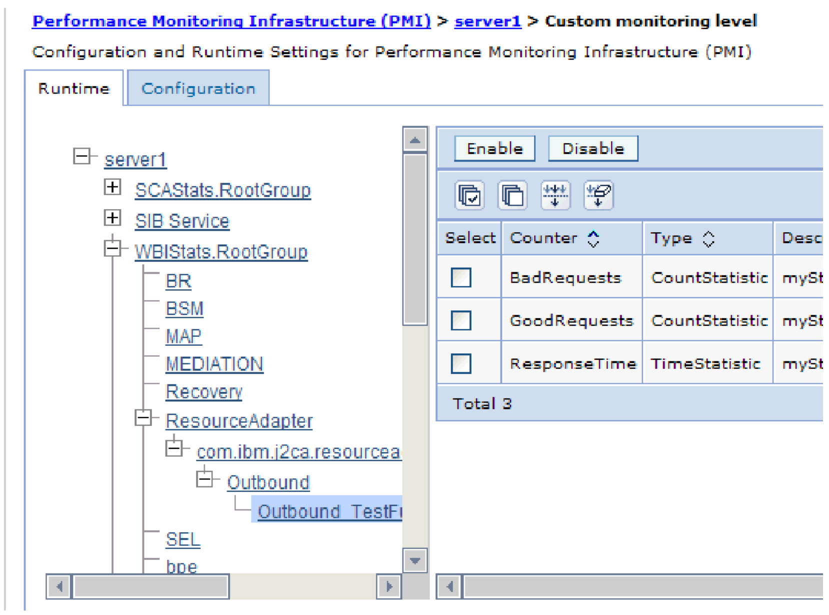
○ Extended
⊞ Provides extended monitoring (Basic + WebSphere components)

○ All
⊞ All statistics are enabled

◉ Custom
Provides fine-grained control to selectively enable statistics

Figure 31. Enabling Performance Monitoring Infrastructure

   e. Click **Apply** or **OK**.
   f. Click **Save**. PMI is now enabled.
2. Configure PMI for your adapter.
   a. In the administrative console, expand **Monitoring and Tuning**, and then select **Performance Monitoring Infrastructure (PMI)**.
   b. From the list of servers, click the name of your server.
   c. Select **Custom**.
   d. Select the **Runtime** tab. The following figure shows the Runtime tab.


Performance Monitoring Infrastructure (PMI) > server1 > Custom monitoring level

Configuration and Runtime Settings for Performance Monitoring Infrastructure (PMI)

Select	Counter	Type	Desc
☐	BadRequests	CountStatistic	mySt
☐	GoodRequests	CountStatistic	mySt
☐	ResponseTime	TimeStatistic	mySt

Total 3

Figure 32. Runtime tab used for configuring PMI

e. Click **WBIStats.RootGroup**. This is a PMI submodule for data collected in the root group. This example uses the name WBIStats for the root group.
f. Click **ResourceAdapter**. This is a submodule for the data collected for the JCA adapters.
g. Click the name of your adapter, and select the processes you want to monitor.
h. In the right pane, select the check boxes for the statistics you want to gather, and then click **Enable**.

Results

PMI is configured for your adapter.

What to do next

Now you can view the performance statistics for your adapter.

Viewing performance statistics

You can view adapter performance data through the graphical monitoring tool, Tivoli Performance Viewer. Tivoli Performance Viewer is integrated with the administrative console in WebSphere Process Server.

Before you begin

Configure Performance Monitoring Infrastructure for your adapter.

Procedure

1. In the administrative console, expand **Monitoring and Tuning**, expand **Performance Viewer**, then select **Current Activity**.
2. In the list of servers, click the name of your server.
3. Under your server name, expand **Performance Modules**.

4. Click **WBIStatsRootGroup**.
5. Click **ResourceAdapter** and the name of your adapter module.
6. If there is more than one process, select the check boxes for the processes whose statistics you want to view.

Results

The statistics are displayed in the right panel. You can click **View Graph** to view a graph of the data, or **View Table** to see the statistics in a table format. The following figure shows adapter performance statistics as a graph.

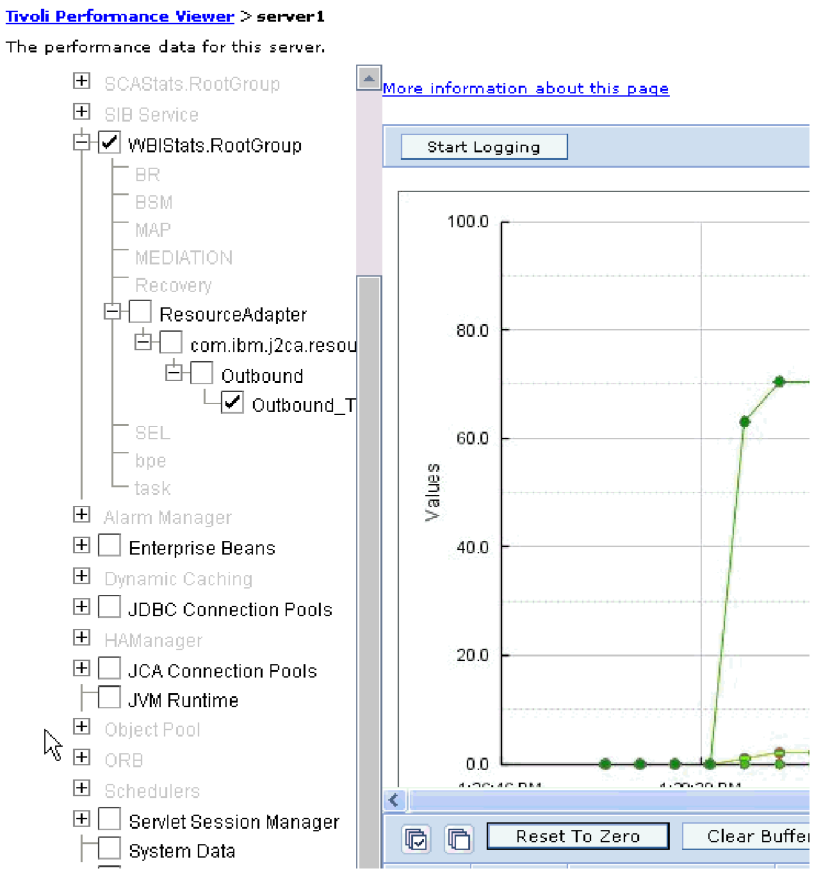


Figure 33. Adapter performance statistics, using graph view

Enabling tracing with the Common Event Infrastructure (CEI)

The adapter can use the Common Event Infrastructure, a component embedded in the server, to report data about critical business events such as the starting or stopping of a poll cycle. Event data can be written to a database or a trace log file depending on configuration settings.

Procedure

1. In the administrative console, click **Troubleshooting**.
2. Click **Logs and Trace**.
3. In the list of servers, click the name of your server.

4. In the **Change Log Detail Levels** box, click the name of the CEI database (for example, WBIEventMonitor.CEI.ResourceAdapter.*) or the trace log file (for example, WBIEventMonitor.LOG.ResourceAdapter.*) to which you want the adapter to write event data.
5. Select the level of detail about business events that you want the adapter to write to the database or trace log file, and (optionally) adjust the granularity of detail associated with messages and traces.
   - **No Logging**. Turns off event logging.
   - **Messages Only**. The adapter reports an event.
   - **All Messages and Traces**. The adapter reports details about an event.
   - **Message and Trace Levels**. Settings for controlling the degree of detail the adapter reports about the business object payload associated with an event. If you want to adjust the detail level, choose one of the following:

     Fine. The adapter reports the event but none of the business object payload.

     Finer. The adapter reports the event and the business object payload description.

     Finest. The adapter reports the event and all of the business object payload.
6. Click **OK**.

Results

Event logging is enabled. You can view CEI entries in the trace log file or by using the Common Base Event Browser within the administrative console.

Troubleshooting and support

Common troubleshooting techniques and self-help information help you identify and solve problems quickly.

Configuring logging and tracing

Configure logging and tracing to suit your requirements. Enable logging for the adapter to control the status of event processing. Change the adapter log and trace file names to separate them from other log and trace files.

Configuring logging properties

Use the administrative console to enable logging and to set the output properties for a log, including the location, level of detail, and output format of the log.

About this task

Before the adapters can log monitored events, you must specify the service component event points that you want to monitor, what level of detail you require for each event, and format of the output used to publish the events to the logs. Use the administrative console to perform the following tasks:
- Enable or disable a particular event log
- Specify the level of detail in a log
- Specify where log files are stored and how many log files are kept
- Specify the format for log output

  If you set the output for log analyzer format, you can open trace output using the Log Analyzer tool, which is an application included with your process server. This is useful if you are trying to correlate traces from two different server processes, because it allows you to use the merge capability of the Log Analyzer.

For more information about monitoring on a process server, including service components and event points, see the documentation for your process server.

You can change the log configuration statically or dynamically. Static configuration take effect when you start or restart the application server. Dynamic, or runtime, configuration changes apply immediately.

When a log is created, the detail level for that log is set from the configuration data. If no configuration data is available for a particular log name, the level for that log is obtained from the parent of the log. If no configuration data exists for the parent log, the parent of that log is checked, and so on up the tree, until a log with a non-null level value is found. When you change the level of a log, the change is propagated to the children of the log, which recursively propagate the change to their children, as necessary.

To enable logging and set the output properties for a log, use the following procedure.

Procedure

1. In the navigation pane of the administrative console, click **Servers** → **Application Servers**.
2. Click the name of the server that you want to work with.
3. Under **Troubleshooting**, click **Logs and trace**.
4. Click **Change Log Detail Levels**.
5. Specify when you want the change to take effect:
   - For a static change to the configuration, click the **Configuration** tab.
   - For a dynamic change to the configuration, click the **Runtime** tab.
6. Click the names of the packages whose logging level you want to modify. The package names for WebSphere Adapters start with **com.ibm.j2ca**:
   - For the adapter base component, select **com.ibm.j2ca.base**.
   - For the adapter base component and all deployed adapters, select **com.ibm.j2ca.base.***.
   - For the Adapter for JDBC only, select the **com.ibm.j2ca.jdbc** package.
7. Select the logging level.

Logging Level	Description
Fatal	The task cannot continue or the component cannot function.
Severe	The task cannot continue, but the component can still function. This logging level also includes conditions that indicate an impending fatal error, that is, situations that strongly suggest that resources are on the verge of being depleted.
Warning	A potential error has occurred or a severe error is impending. This logging level also includes conditions that indicate a progressive failure, for example, the potential leaking of resources.
Audit	A significant event has occurred that affects the server state or resources.

Logging Level	Description
Info	The task is running. This logging level includes general information outlining the overall progress of a task.
Config	The status of a configuration is reported or a configuration change has occurred.
Detail	The subtask is running. This logging level includes general information detailing the progress of a subtask.

8. Click **Apply**.
9. Click **OK**.
10. To have static configuration changes take effect, stop and then restart the process server.

Results

Log entries from this point forward contain the specified level of information for the selected adapter components.

Changing the log and trace file names

To keep the adapter log and trace information separate from other processes, use the administrative console to change the file names. By default, log and trace information for all processes and applications on a process server is written to the SystemOut.log and trace.log files, respectively.

Before you begin

You can change the log and trace file names at any time after the adapter module has been deployed to an application server.

About this task

You can change the log and trace file names statically or dynamically. Static changes take effect when you start or restart the application server. Dynamic or run time changes apply immediately.

Log and trace files are in the *install_root*/profiles/*profile_name*/logs/*server_name* folder.

To set or change the log and trace file names, use the following procedure.

Procedure

1. In the navigation pane of the administrative console, select **Applications > Enterprise Applications**.
2. In the Enterprise Applications list, click the name of the adapter application. This is the name of the EAR file for the adapter, but without the .ear file extension. For example, if the EAR file is named Accounting_OutboundApp.ear, then click **Accounting_OutboundApp**.
3. In the Configuration tab, in the Modules list, click **Manage Modules**.
4. In the list of modules, click IBM WebSphere Adapter for JDBC.
5. In the Configuration tab, under Additional Properties, click **Resource Adapter**.
6. In the Configuration tab, under Additional Properties, click **Custom properties**.
7. In the Custom Properties table, change the file names.

a. Click either **logFilename** to change the name of the log file or **traceFilename** to change the name of the trace file.
b. In the Configuration tab, type the new name in the **Value** field. By default, the log file is called SystemOut.log and the trace file is called trace.log.
c. Click **Apply** or **OK**. Your changes are saved on your local machine.
d. To save your changes to the master configuration on the server, use one of the following procedures:
   - **Static change**: Stop and restart the server. This method allows you to make changes, but those changes do not take effect until you stop and start the server.
   - **Dynamic change**: Click the **Save** link in the Messages box above the Custom properties table. Click **Save** again when prompted. This method allows you to make changes that take effect right away.

First-failure data capture (FFDC) support

The adapter supports first-failure data capture (FFDC), which provides persistent records of failures and significant software incidents that occur during run time in WebSphere Process Server or WebSphere Enterprise Service Bus.

The FFDC feature runs in the background and collects events and errors that occur at run time. The feature provides a means for associating failures to one another, allowing software to link the effects of a failure to their causes, and thereby facilitate the quick location of the root cause of a failure. The data that is captured can be used to identify exception processing that occurred during the adapter run time.

When a problem occurs, the adapter writes exception messages and context data to a log file, which is located in the *install_root*/profiles/*profile*/logs/ffdc directory.

For more information about first-failure data capture (FFDC), see the WebSphere Process Server or WebSphere Enterprise Service Bus documentation.

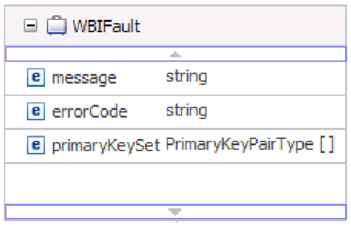
Business faults

The adapter supports business faults, which are exceptions that are anticipated and declared in the outbound service description, or import. Business faults occur at predictable points in a business process as a result of a business rule violation or a constraint violation.

Although WebSphere Process Server and WebSphere Enterprise Service Bus support other types of faults, the adapter generates only business faults, which are called simply *faults* in this documentation. Not all exceptions become faults. Faults are generated for errors that are actionable, that is, errors that can have a recovery action that does not require the termination of the application. For example, the adapter generates a fault when it receives a business object for outbound processing that does not contain the required data or when the adapter encounters certain errors during outbound processing.

Fault business objects

The external service wizard creates a business object for each fault that the adapter can generate. In addition, the wizard creates a WBIFault superset business object, which has information common to all faults, such as the message, errorCode, and primarySetKey attributes as shown in Figure 34 on page 150.



message	string
errorCode	string
primaryKeySet	PrimaryKeyPairType [ ]

Figure 34. The structure of the WBIFault business object

Some faults contain the matchCount attribute, to provide additional information about the error. For others, WBIFault contains all the information needed to handle the fault.

The wizard creates the following fault business objects:

- IntegrityConstraintFault

  When processing a Create operation, the adapter throws an integrity constraint fault if the database throws the SQLException exception for an integrity constraint violation. For example, if a foreign key is not found, the adapter throws this fault.
- MatchesExceededLimitFault

  When processing the processing of an RetrieveAll operation, the adapter throws this fault if the number of records returned from the database query exceed the maximum number of records property in the interaction specification.

  To increase the number of records that can be returned, increase the value of the MaxRecords property in the interaction specification properties for the RetrieveAll operation.

  The business object for this fault has one property, matchCount, which is a string that contains the number of matches.
- MissingDataFault

  If the business object that is passed to the outbound operation does not have all the required attributes, then the adapter throws this fault.
- MultipleMatchingRecordsFault

  When processing a Retrieve operation, the adapter throws this fault if the query returns more than one record for the keys specified. The business object for this fault has one property, matchCount, which is a string that contains the number of matches.
- ObjectNotFoundFault

  When processing the Create and Update operations, the adapter retrieves the single-cardinality child object if the ownership is false for this child object. If the retrieval does not return anything, then the adapter throws this fault.
- RecordNotFoundFault

  When processing a data retrieval operation, the adapter throws this fault if the record is not found in the database for the keys specified. This fault can occur for the Delete, Update, Retrieve, and RetrieveAll operations.
- UniqueConstraintFault

  When processing a Create operation, the adapter throws this fault if it receives an SQLException exception from the database because of a unique constraint violation.

Configuring the module for fault processing

Before you can configure your module to support business faults, you must have used the external service wizard to configure your module.

To enable fault processing, you must modify the .import and WSDL files for your module. You can configure faults at either the binding level or the method level. If the changes are made at binding level, they apply to all methods in the import. If the changes are made at the method binding level, you can configure a different fault for each method.

Table 13 lists the fault name and fault binding for each fault. Use the fault name and fault binding class when you configure the module.

Table 13. The fault name and fault binding class for each fault

Fault name	Associated fault binding class
INTEGRITY_CONSTRAINT_VIOLATION	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl
MATCHES_EXCEEDED_LIMIT	com.ibm.j2ca.extension.emd.runtime.MatchingFaultDataBinding
MISSING_DATA	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl
MULTIPLE_MATCHING_RECORDS	com.ibm.j2ca.extension.emd.runtime.MatchingFaultDataBinding
OBJECT_NOTFOUND_EXCEPTION	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl
RECORD_NOT_FOUND	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl
UNIQUECONSTRAINT_VIOLATION	com.ibm.j2ca.extension.emd.runtime.WBIFaultDataBindingImpl

1. Edit the .import file to configure the fault at either the binding or the method level.
   - To configure the faults at the binding level:
     a. In the binding section, add the faultSelector attribute and the name of the fault selector. The name of the fault selector is com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl.
     b. For each fault that you want to enable, add a <faultBinding> element. In the element, specify the fault name and the fault data binding class name from Table 13.

        The following .import file shows the MULTIPLE_MATCHING_RECORDS and RECORD_NOT_FOUND faults configured for all methods. **Bold face type** indicates changes made to enable fault handling.

```
<esbBinding xsi:type="eis:EISImportBinding"
dataBindingType="com.ibm.j2ca.jdbc.emd.databinding.JDBCDataBindingGenerator"
 faultSelector="com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl">
    <resourceAdapter name="JDBCXAApp.IBM WebSphere Adapter for JDBC"
type="com.ibm.j2ca.jdbc.JDBCResourceAdapter" version="6.1">
       <properties>
         <adapterID>CWYBC_JDBC</adapterID>
       </properties>
    </resourceAdapter>
    <faultBinding fault="MULTIPLE_MATCHING_RECORDS"
          faultBindingType="com.ibm.j2ca.extension.emd.runtime.MatchingFa
ultDataBinding"/>
    <faultBinding fault="RECORD_NOT_FOUND"
          faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultDa
taBindingImpl"/>
```

   - To configure the faults at the method level:
     a. In method binding section for the method you want to associate with the fault, add the name of the fault selector. The value for fault selector is com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl.

b. Add the fault binding elements in the method binding section. Use the fault name and the corresponding fault data binding class name from Table 13 on page 151.

The following .import file shows the MULTIPLE_MATCHING_RECORDS and RECORD_NOT_FOUND faults configured for only the retrieveRtasserCustomerBG method. **Bold face type** indicates changes made to enable fault handling.

```
<methodBinding inDataBindingType="com.ibm.xmlns.prod.websphere.j2ca.jdbc.r
tassercustomerbg.RtasserCustomerBGDataBinding"
   method="retrieveRtasserCustomerBG"
   outDataBindingType="com.ibm.xmlns.prod.websphere.j2ca.jdbc.rtassercustom
erbg.RtasserCustomerBGDataBinding"
   faultSelector="com.ibm.j2ca.extension.emd.runtime.WBIFaultSelectorImpl">
      <interaction>
        <properties>
          <functionName>Retrieve</functionName>
        </properties>
      </interaction>
      <faultBinding fault="MULTIPLE_MATCHING_RECORDS"
         faultBindingType="com.ibm.j2ca.extension.emd.runtime.MatchingFau
ltDataBinding"/>
      <faultBinding fault="RECORD_NOT_FOUND"
         faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultDataB
indingImpl"/>
</methodBinding>
```

2. Determine the target namespaces for your faults. For each fault that you want to enable, determine the namespace as follows:
   a. Open the fault schema (XSD file) in a text editor.
   b. Locate the target namespace. The target namespace is shown in **bold face type** in the following portion of a fault schema:

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://com/ibm/j2ca/fault/afcfault"
   xmlns:basefault="http://com/ibm/j2ca/fault">
<import namespace="http://com/ibm/j2ca/fault" schemaLocation="WBIFault.xsd"/>

. . .
```

   The faults can all have the same target namespace or they can have different target namespaces.

3. Edit the WSDL file to declare the faults for the service. A sample WSDL file with these changes made is shown at the end of the list.
   a. In the <definitions> element, add a namespace for each fault namespace, using the information you obtained from the fault schema files. If all your fault schemas have the same targetNamespace, add only one alias. If they have different targetNamespaces, add an alias for each unique namespace.
   b. Create an <xsd:import> element to import the schema for each fault you want to enable.
   c. Declare import statements for each fault type. Make sure that you are using the correct alias defined in step 3a to resolve the complex type in `type=`*`alias`*`:`*`faultBOName`*`.xsd`.
   d. Declare the message tags for each of the fault types.
   e. Add the fault declaration to each method where faults should be handled.

The following WSDL file defines the MULTIPLE_MATCHING_RECORDS and RECORD_NOT_FOUND faults. **Bold face type** indicates changes made to enable fault handling.

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:RtasserCustomerBG="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtassercusto
merbg"
  xmlns:RtasserCustomerContainer="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtass
ercustomercontainer"
  xmlns:intf="http://JDBCXA/JDBCOutboundInterface"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

Step 3a on page 152

```
  xmlns:fault="http://com/ibm/j2ca/fault/afcfault"
     name="JDBCOutboundInterface.wsdl"
     targetNamespace="http://JDBCXA/JDBCOutboundInterface">
  <types>
<xsd:schema
  xmlns:tns="http://JDBCXA/JDBCOutboundInterface"
  xmlns:xsd1="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtassercustomerbg"
  xmlns:xsd2="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtassercustomercontainer"
  elementFormDefault="qualified"
  targetNamespace="http://JDBCXA/JDBCOutboundInterface"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:import
   namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtassercustomerbg"
   schemaLocation="RtasserCustomerBG.xsd"/>
<xsd:import
   namespace="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/rtasseracustomercontainer"
   schemaLocation="RtasserCustomerContainer.xsd"/>
```

Step 3b on page 152

```
<xsd:import namespace="http://com/ibm/j2ca/fault/afcfault"
   schemaLocation="MultipleMatchingRecordsFault.xsd"/>
<xsd:import namespace="http://com/ibm/j2ca/fault/afcfault"
   schemaLocation="RecordNotFoundFault.xsd"/>

. . .
```

Step 3c on page 152

```
     <xsd:element name="multipleMatchingRecordsFaultX">
      <xsd:complexType>
       <xsd:sequence>
       <xsd:element name="multipleMatchingRecordsFaultElement"
            type="fault:MultipleMatchingRecordsFault"/>
        </xsd:sequence>
      </xsd:complexType>
     </xsd:element>

     <xsd:element name="recordNotFoundFaultX">
       <xsd:complexType>
        <xsd:sequence>
        <xsd:element name="recordNotFoundFaultElement"
             type="fault:RecordNotFoundFault"/>
         </xsd:sequence>
       </xsd:complexType>
     </xsd:element>
    </xsd:schema>
  </types>

  . . .
```

Step 3d on page 152

```
<message name="multipleMatchingRecordsFault">
  <part element="intf:multipleMatchingRecordsFaultX"
        name="multipleMatchingRecordsFaultPart"/>
</message>
<message name="recordNotFoundFault">
  <part element="intf:recordNotFoundFaultX"
        name="recordNotFoundFaultPart"/>
</message>
<portType name="JDBCOutboundInterface">

. . .

<operation name="retrieveRtasserCustomerBG">
<input message="intf:retrieveRtasserCustomerBGRequest"
       name="retrieveRtasserCustomerBGRequest"/>
<output message="intf:retrieveRtasserCustomerBGResponse"
        name="retrieveRtasserCustomerBGResponse"/>
```

Step 3e on page 152

```
<fault message="intf:multipleMatchingRecordsFault"
     name="multipleMatchingRecordsFaultFault" />
<fault message="intf:recordNotFoundFault"
     name="recordNotFoundFaultFault" />
</operation>
</portType>
</definitions>
```

XAResourceNotAvailableException

When the process server log contains repeated reports of the com.ibm.ws.Transaction.XAResourceNotAvailableException exception, remove transaction logs to correct the problem.

Symptom:

When the adapter starts, the following exception is repeatedly logged in the process server log file:

com.ibm.ws.Transaction.XAResourceNotAvailableException

Problem:

A resource was removed while the process server was committing or rolling back a transaction for that resource. When the adapter starts, it tries to recover the transaction but cannot because the resource was removed.

Solution:

To correct this problem, use the following procedure:

1. Stop the process server.
2. Delete the transaction log file that contains the transaction. Use the information in the exception trace to identify the transaction. This prevents the server from trying to recover those transactions.

   Note: In a test or development environment, you can generally delete all of the transaction logs. In WebSphere Integration Developer, delete the files and subdirectories of the transaction log directory, *server_install_directory*\profiles*profile_name*\tranlog.

   In a production environment, delete only the transactions that represent events that you do not need to process. One way to do this is to reinstall the adapter, pointing it to the original event database used, and deleting only the transactions you do not need. Another approach is to delete the transactions from either the log1 or log2 file in the following directory:

server_install_directory\profiles*profile_name*\tranlog*node_name*\wps\ *server_name*\transaction\tranlog

3. Start the process server.

Self-help resources

Use the resources of IBM software support to get the most current support information, obtain technical documentation, download support tools and fixes, and avoid problems with WebSphere Adapters. The self-help resources also help you diagnose problems with the adapter and provide information about how to contact IBM software support.

Support Web site

The WebSphere Adapters software support Web site at http://www.ibm.com/software/integration/wbiadapters/support/ provides links to many resources to help you learn about, use, and troubleshoot WebSphere Adapters, including the following types of

- Flashes (alerts about the product)
- Technical information including the product information center, manuals, IBM Redbooks®, and whitepapers
- Educational offerings
- Technotes

Recommended fixes

A list of recommended fixes you should apply is available at the following location: http://www.ibm.com/support/docview.wss?fdoc=aimadp&rs=695 &uid=swg27010397

Technotes

Technotes provide the most current documentation about the Adapter for JDBC, including the following topics:

- Problems and their currently available solutions
- Answers to frequently asked questions
- How-to information about installing, configuring, using, and troubleshooting the adapter
- *IBM Software Support Handbook*

For a list of technotes for WebSphere Adapters, visit this address:

http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8 &dc=DB520+D800+D900+DA900+DA800+DB560&dtm

Plug-in for IBM Support Assistant

Adapter for JDBC provides a plug-in for IBM Support Assistant, which is a free, local software serviceability workbench. For information about installing or using IBM Support Assistant, visit this address:

http://www.ibm.com/software/support/isa/

Solutions to common problems

Some of the problems you may encounter running WebSphere Adapter for JDBC with your database are described, along with solutions and workarounds. These problems and solutions are in addition to those documented as technotes on the software support Web site.

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

RecordNotFoundException for RetrieveAll operation in test client

Problem

When performing a RetrieveAll operation in the WebSphere Integration Developer test client, the RecordNotFoundException exception is generated when data is expected from the query. The following message is generated: `RecordNotFoundException: Record not found in EIS`.

Cause

This except can occur if the WHERE clause for the SELECT statement does not set all of the attributes of the business object. Leaving the attribute blank, which is the default value, is not the same as explicitly unsetting the value.

Solution

In the test client, set the values of the attributes that are required to `<unset>`. Repeat the RetrieveAll operation. If the exception is generated again, it is likely that no matching records exist in the database table.

CLOB datatypes of 4K or larger cannot be inserted into Oracle 9i or 10g databases

Problem

You get the following exception when inserting CLOB (character large object) values of 4K and larger into Oracle 9i or 10g databases:

- Oracle 9i: `ResourceAdapt E com.ibm.j2ca.jdbc.JDBCDBOperationHandler executePreparedCUDStatement CWYBC0301E: An operation on the database failed with a SQL exception for the following reason: No more data to read from socket.`
- Oracle 10g: `ResourceAdapt E com.ibm.j2ca.jdbc.JDBCDBOperationHandler executePreparedCUDStatement CWYBC0301E: An operation on the database failed with a SQL exception for the following reason: ORA-01460: unimplemented or unreasonable conversion requested`

Cause

You are using an older driver that does not correctly support CLOBs larger than 4K.

Solution

Use the Oracle thin driver from Oracle 10.1.0.2 or a later release.

Some generated business objects have no attributes for Oracle database objects

Problem

For some business objects that are generated from an Oracle database object, the generated business object has no attributes for the table columns.

Cause

Under certain conditions, the Oracle JDBC driver does not return the column information for a database object. The following bugs are currently filed with Oracle for these issues:

- 2281705. DATABASEMETADA.GETCOLUMNS does not return underlying table if there is a synonym
- 2696213. JDBC GETPROCEDURECOLUMNS does not return columns for the synonym of a procedure

Additionally, column information will not be returned if a private synonym that references an object in another schema is used.

Solution

For tables that have a synonym, generate the business object using the synonym for the table.

For synonyms of a procedure, generate the business object using the original procedure that the synonym is based on.

For private synonyms that reference an object in another schema, either use the original table or create a synonym in the current schema.

ResourceException exceptions during outbound processing

If you get a ResourceException exception, examine the root cause field to determine the cause. Common problems have the following root causes:

- SQLException exception

  If the SQLException exception includes the text `UserID or password is invalid`, then the user ID or password specified for the outbound connection are not correct.

  For example:

  ```
  javax.resource.ResourceException: [ibm][db2][jcc][t4][2013][11249] Connection
  authorization failure occurred.  Reason: User ID or Password invalid.
  ```
- ConnectException exception

  If the text included with the ConnectException exception includes text similar to `connection refused` or `could not establish connection to the server`, then the database server might not be operational or there might be a network problem that prevents a connection.

  For example:

  ```
  javax.resource.ResourceException: [ibm][db2][jcc][t4][2043][11550] Exception
  java.net.ConnectException: Error opening socket to server /9.26.237.55 on port
  50,000 with message: Connection refused: connect.
  ```

ResourceException exception during inbound processing

This exception indicates that there is a repeated problem connecting to the database. To polls for events, the adapter must connect top the database. If the connection fails, the adapter waits a configurable amount of time before trying to connect again. The adapter retries a configurable number of times before it stops polling. When the adapter stops polling, it generates the ResourceException exception.

UniqueConstraintViolation fault, MultiMatchingRecordsException fault

Class loader violation occurs when starting the external service wizard

Problem

It is not possible to use the external service wizard after using a connection to the database in the Data perspective. At the end of the second panel of the wizard, the following exception is generated:

com.ibm.adapter.framework.api.ImportException
Reason: class loading constraint violated (class:
oracle/jdbc/driver/OracleConnection
method: getWrapper()Loracle/jdbc/OracleConnection;) at pc:0

The error occurs in both of the following situations:

- When you establish a connection to the database through the external service wizard, an error occurs when you attempt to connect to the database from the Data perspective.
- When you establish a connection to the database through the Data perspective, the error occurs when you attempt to connect to database through the external service wizard.

Cause

The error occurs because the Data perspective and the wizard use their own class loaders. Once the DLL, which is the native library used by the JDBC driver, is loaded in the Data perspective, it cannot be loaded again in the wizard. JVMs have an inherent restriction that only allows one class loader to load native libraries at any given time. So if class loader A loads DLL B, then no other class loader can load DLL B until class loader A is released and garbage is collected. Because you cannot really control garbage collection, it usually means that if you want to load DLL B with another class loader, you need to restart the JVM. This limitation is a known one and is documented for WebSphere Application Server.

Solution

The only solution is to restart WebSphere Integration Developer when this error occurs.

Closed connection error occurs when using XA with Oracle 10g

Problem

When the Adapter for JDBC is used to perform an XA transaction using Oracle 10g, the adapter generates a closed connection exception: javax.resource.ResourceException: Closed Connection.

Cause

This is a known issue with the Oracle 10g database driver. The following bug has been filed with Oracle for this issue: 3488761 Connection closed error from OracleConnection.getConnection() - 10G drivers.

Solution

The bug has been fixed in the Oracle 10g Release 2 driver. As a workaround, you can use the Oracle 9i JDBC thin drivers to connect to the database for XA transactions.

Error occurs while starting a transaction on Oracle

Problem

When the Adapter for JDBC is used to perform an XA transaction using Oracle database, the following error is generated: `WTRN0078E: An attempt by the transaction manager to call start on a transactional resource has resulted in an error. The error code was XAER_RMERR.`

Cause

For the Oracle database server to support XA transactions, some commands need to be run.

Solution

Two scripts that are included in the Oracle directory should be run. This activity will likely need to be performed by your Oracle database administrator, because you must be logged into Oracle as SYSOPER or SYSDBA in order to have the necessary permissions for these scripts to work. The scripts are:

```
<ORACLE_HOME>javavm\install
file: initxa.sql
file: initjvm.sql
```

The initxa.sql script configures the database for XA. Once it runs successfully, your database is configured for XA. The script might run successfully the first time you try. Unfortunately, it probably will not run successfully because some of the database's memory spaces are too small.

To fix this, run the initjvm.sql script. It will probably fail too, but in doing so, it will indicate which parameters need to be adjusted. The parameters are stored in this file:

```
<ORACLE_HOME>\database
file: init<DATABASE_SID>.ora
```

Table 14 on page 160 shows two parameters that typically need to be increased. Your particular database configuration might require adjustment of different parameters.

Table 14. Typical parameter sizes

Parameter Name	Minimum Value
java_pool_size	12000000
shared_pool_size	24000000

Using the adapter to connect to IBM DB2 for z/OS with a JDBC (Type 2 or Type 4) universal driver

Problem and Cause

DB2 for z/OS supports querying stored procedure metadata by using positional index by default and not using column name, which the Adapter for JDBC uses. The solution provides the steps for using the Adapter for JDBC with DB2 on the z/OS platform.

Solution

To connect to DB2 for z/OS using the Adapter for JDBC, ensure that the following connection requirements are met:

- The physical representation of the universal JDBC driver is the db2jcc.jar file. Ensure that the path to this file is set in the class path.
- Database URL: To determine whether you are using the Type 2 or Type 4 driver, review the form of the connection:

  Type 2: jdbc:db2:database
  (For example: jdbc:db2:MyDB, where MyDB is the database name)

  Type 4: jdbc:db2://server:port/database
  (For example: jdbc:db2://9.182.15.129:50000/MyDB,, where MyDB is the database name)
- Driver class: com.ibm.db2.jcc.DB2Driver.

  The driver class for both Type 2 and Type 4 drivers is the same.
- Set the path of the db2jcc_license_cisuz.jar file in the class path.

  The license JAR file is the same for both Type 2 and Type 4 drivers. Access to DB2 for z/OS and DB2 for i5/OS servers requires a valid DB2 Connect™ license. DB2 clients do not provide direct connectivity to zSeries® and iSeries® servers without a DB2 Connect license.

  For more information on DB2 Connect licensing and usage, refer to the following pages:
  http://www-128.ibm.com/developerworks/db2/library/techarticle/0303zikopoulos1/0303zikopoulos1.html
  http://www-128.ibm.com/developerworks/db2/library/techarticle/0301zikopoulos/0301zikopoulos.html

There might be issues with importing metadata for stored procedures using the wizard. To use stored procedures and import metadata from DB2 using the Adapter for JDBC, DB2 needs to be reconfigured as described in the following steps. Follow these steps in addition to the steps provided above:

- Apply the following APARs on DB2: PQ62695, PQ55393, PQ56616, PQ54605, PQ46183, and PQ62139.
- If you want to use stored procedures with the adapter, follow the steps below, which are part of the fix for PQ62695. This fix introduces stored procedures that

provide the ability to generate a result set that corresponds to the Schema Metadata APIs documented in the JDBC and ODBC specification.

These procedures will be used by the JDBC and ODBC drivers provided in the DB2 Universal Driver. Follow these steps to enable support for stored procedures:

1. Apply the APAR.
2. Check the value of the DESCSTAT variable in the ZPARM assembly job DSNTIJUZ. If the value of the DESCSTAT variable is NO, change it to YES.

   Note: The default for DESCSTAT is NO on V7 but was changed to YES on V8.
3. Reassemble and re-initialize the ZPARM module.
4. Run the JCL job named DSNTIJMS. You can find this member in the db2prefix.SDSNSAMP data set.
5. Restart DB2.

Using XA transactions for outbound support with a remote DB2 database

This provides the steps, database versions, and configuration requirements for XA transaction support using the Adapter for JDBC with a remote DB2 database.

Using XA Transactions with the Adapter for JDBC using the Universal Driver

The following versions of software and configuration properties are needed to use XA transactions with the Adapter for JDBC and the Universal driver to connect to a remote DB2 database:

- DB2 version: 8.2 or later
- JDBC Driver: UDB driver (db2jcc.jar) Type 4
- XA datasource name: com.ibm.db2.jcc.DB2XADataSource
- XA Database name: This is the remote database alias configured on the local DB2 client.
- Database URL: jdbc:db2://hostname:port/databasename
- JDBC driver class: com.ibm.db2.jcc.DB2Driver

Using XA Transactions on a remote DB2 database

Adding a remote DB2 database

1. Run the db2admin (*DB2_InstallPath*\SQLLIB\BIN) command on the DB2 server.
2. Open the DB2 Configuration Assistant.
3. Go to **View** → **Advanced View**.

Complete the following steps, in order:

1. **Add the remote system**
   a. Select **Systems** tab.
   b. From the menu, choose **Selected** → **Add System** .
   c. In the **System name** field, specify the physical machine, server system, or workstation where the target database is located. The system name on the server system is defined by the DB2SYSTEM DAS configuration parameter. This is the value that you should use.

   d. In the **Host name** field, type the host name or Internet Protocol (IP) address where the target database resides.
   e. In the **Node name** field, specify a local nickname for the remote node where the database is located. The node name you choose must not already exist in the node directory or the admin node directory.
   f. Select the operating system and click **OK**.
2. **Add an instance node**
   a. Select the **Instance Nodes** tab.
   b. From the menu, choose **Selected** → **Add Instance Node** .
   c. In the **System name** field, specify the physical machine, server system, or workstation where the target database is located. Select the system added in the Adding a remote system task.
   d. In the **Instance name** field, type the name of the instance (DB2, and so on) where the target database is located.
   e. In the **Instance node name** field, specify a unique nickname for the cataloged system (node) where the database is located. The node name you choose must not already exist in the node directory or the admin node directory.
   f. Select the operating system and type the host name. Use the same host name as in step 4 of the task: Adding the remote system.
   g. Enter the port number on which the remote DB2 instance is running.
   h. Click **OK**.
3. **Add a database**
   a. Select the **Databases** tab.
   b. From the menu, choose **Selected** → **Add Database**.
   c. In the **Instance node** field, choose the instance created in the task: Adding an instance node. Specify the name of the database that you are adding in the **Database name** field.
   d. In the **Alias** field, specify a local nickname that can be used by applications running on your workstation. If nothing is entered, the alias will be the same as the database name. The alias name should be unique.

      Note: This alias name value should be entered in the XADatabaseName property for the adapter.
4. **Test the database connection**
   a. Select the **Databases** tab.
   b. Choose the database added in the task: Adding a database.
   c. From the menu, choose **Selected** → **Test Connection**.
   d. Select the **CLI** check box, type the User Id and Password and click **Test Connection**. This should return a successful connection.

A closer look at the transaction (XID) column in the event table

If the adapter is configured for assured once delivery, use the status column with the XID column to determine whether the event has been processed:

- If the XID column contains 0, the event has not yet been picked up for processing.
- If the XID column contains a transaction ID (that is, it does not contain 0), then the adapter has started to process the event but has not completed processing. You might see this combination when the adapter or application server crashes

while the event is bring processed. The transaction manager will either COMMIT or ROLLBACK these transactions during recovery.

Chapter 8. Reference

2 Detailed information about business objects, adapter properties (enterprise service
2 discovery properties, resource adapter properties, managed (J2C) connection
2 factory properties, activation specification properties, and interaction specification
2 properties), messages, and related product information is provided for your
2 reference.

Business object information

A business object is a structure that contains application-specific information (metadata) about how the adapter should process the business object as well as the operation to be performed on the business object. The name of the business object is generated by the external service wizard in accordance with the naming convention for the adapter.

Business object attributes

Business object attributes define the content of a business object and are built from the list of columns in the database object. Each attribute has a name, type, cardinality, and several other properties. The external service wizard sets the attribute name to the name of the column. The adapter adds the attribute cardinality, type, and application-specific information.

A business object is simply a container for the data specified in the attributes. The structure of the data in the database is defined by the business object, but data in the database is in the business object attributes.

Table 15 lists the properties of a business object attribute and describes their interpretation and settings.

Table 15. Attribute properties

Properties	Interpretation and settings
Cardinality	An integer specifying the cardinality of a business object. Each business object attribute that represents a child or an array of child business objects has the value of single (1) or multiple (an unbounded integer) cardinality, respectively. In both single- and multiple-cardinality relationships, 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.
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. The RetrieveAll operation overrides the use of keys and foreign keys. **Note:** The adapter does not support specifying an attribute that represents a child business object as a foreign key.
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.

Table 15. Attribute properties (continued)

Properties	Interpretation and settings
MinOccurs MaxOccurs	If the column is not a primary key and is not nullable, the MinOccurs and MaxOccurs attributes are required, and their values are set to at least 1.
Primary Key	Indicates whether this attribute is a primary key. At least one simple attribute in each business object must be specified as the primary key. If the primary key property is set to `true` for a simple attribute, the adapter adds that attribute to the WHERE clause of the SELECT statement and SQL UPDATE statements that it generates while processing the business object. The RetrieveAll operation overrides the use of primary and foreign keys. **Note:** The adapter does not support specifying an attribute that represents a child business object or an array of child business objects as a primary key attribute.
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	For simple attributes, this property specifies the type of the attribute, such as Integer, String, Date, Timestamp, Boolean, Double, or Float. The supported types for simple attributes and their mapping to the JDBC type of a database object are described in Table 16. For attributes that specify a child business object, this property specifies the name of the business object.

The type of each database object, returned as the JDBC metadata, maps to the business object attribute types as listed in Table 16. Only the JDBC types listed are supported by the adapter. Any columns with types that are not listed are not added to the business object. An informational message is produced explaining the problem, 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`.

Table 16. JDBC metadata column type and business object attribute types

JDBC metadata column type	Business object attribute type
BIT	Boolean
CHAR LONGVARCHAR VARCHAR	String
INTEGER NUMERIC SMALLINT TINYINT BIGINT	Int

Table 16. JDBC metadata column type and business object attribute types (continued)

JDBC metadata column type	Business object attribute type
TIME TIMESTAMP DATE	String
DECIMAL	String
DOUBLE FLOAT	Double
REAL	Float
BLOB	hexBinary
CLOB	String
BINARY VARBINARY LONGBINARY	hexBinary
NCHAR NVARCHAR NTEXT	String
TEXT	String
RAW	String
MONEY SMALLMONEY	String

Attribute application-specific information

Application-specific information (ASI) for business object attributes differs depending on whether the attribute is a simple attribute, or is an attribute that represents a child or an array of child business objects. The application-specific information for an attribute that represents a child 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 only parameter that is required for a simple attribute is the column name. The application-specific information for simple attributes is described in Table 17.

Table 17. Application-specific information for simple attributes

Parameter	Type	Description	Default value
BLOB	Boolean	Indicates whether the database column that corresponds to this attribute has the BLOB data type. While displaying BLOB data, the adapter displays the number of bytes as a hexadecimal value. The attribute type is hexBinary. If `True`, the column data type is BLOB.	None
ByteArray	Boolean	Specifies whether the column is a binary data type. If `True`, the adapter reads and writes binary data to the database and sends that data as a string to the application server. The adapter sets binary data on the business object. The attribute type is hexBinary.	`False`

Table 17. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
ChildBOType	String	If the attribute is a complex data type, use this application-specific information to specify the actual type: • Struct • Array • ResultSet	None
ChildBOTypeName	String	When the value of the ChildBOType application-specific information is either Struct or Array, this parameter value represents the name of the user-defined type. This value is case-sensitive.	
CLOB	Boolean	Indicates whether the database column that corresponds to this attribute has the CLOB data type. This value applies only to attributes of type String. If `True`, the column data type is CLOB. The CLOB attribute has a String Type whose length is used to define the length of the CLOB.	None
ColumnName	String	The name of the database column corresponding to this attribute. This is the only required parameter.	None
CopyAttribute	String	A user-specified value that refers to another attribute name from within the same business object or parent business object. If the value set in the application-specific information refers to the name of another attribute within the same business object, then the adapter uses the value of the other attribute to set the value of this attribute (on which application-specific information is defined) before it adds the business object to the database during a Create operation. For example, if you want the `contact` column of a new row in the table to contain the same value as the `email` column, set the CopyAttribute parameter of the `contact` attribute to `email`. The value cannot reference an attribute in a child business object, but it can reference an attribute in the parent business object by preceding the name with two periods. For example, you can reference the `ccode` attribute in a parent business object as `..ccode`. If you do not include this parameter in the application-specific information, the adapter uses the value of the current attribute without copying the value from another attribute.	None

Table 17. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
DateType	String	Specifies that the corresponding element is a date, time or time stamp. Specify one of the following values: • Date • Time • Timestamp When setting the value of an attribute of the DateType type, use the following formats: • For Date, use yyyy-MM-dd • For Time, use hh:mm:ss • For Timestamp, use yyyy-MM-dd hh:mm:ss	None
FixedChar	Boolean	Specifies whether the attribute is of fixed length when the columns in the table are of type CHAR, not VARCHAR. For example, when set to `true`, 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. This parameter must be updated manually in the XSD file the business object. Use the business object Editor in WebSphere Integration Developer to edit the XSD file. Ensure that no validation errors occur in the XSD file after it has been updated. See the code example for this parameter following this table. An example of how to specify the FixedChar application-specific information in an XSD file follows this table.	`false`
ForeignKey	String	The value of this property depends on whether the parent/child relationship is stored in the parent business object or in the child. If the relationship is stored in the parent, the value includes both the type of the child business object and the name of the attribute in the child to be used as the foreign key *(Child_BO_name/Child_Property_Name)*. If the relationship 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.	None
OrderBy	String	If a value is specified 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 specify the retrieval order.	None
PrimaryKey	Boolean	If the column associated with this attribute is a primary key in the corresponding table in the database, PrimaryKey is to `True`,	None

Table 17. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
SPParameterType	String	Specifies the type of stored procedure Possible values are: • IP (input only) • OP (output only) • IO (input and output) • RS (result set)	None
UniqueIdentifier (UID)	String	The adapter uses this parameter to generate the unique ID for the business object. It supports the generation of sequences and identity columns. The format of this parameter is as follows: UID=AUTO|*Sequence_Name* For a sequence, set the UID attribute to the name of the sequence. Sequences can be defined for DB2 and Oracle databases only. For an identity column, set the UID attribute to AUTO. 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.	None

The format of attribute application-specific information is shown in the following example section of an XSD file:

Example 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>
```

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>
```

Application-specific information for attributes of type child business object

Two application-specific information parameters are used for attributes that refer to child business objects (complex, as opposed to simple, attributes). When you set this application-specific information, specify the parameters listed in Table 18.

Table 18. Application-specific information for attributes of type child business object

Parameter	Type	Description	Default value
KeepRelationship	Boolean	If `True`, this parameter prevents the deletion of a child business object during an Update operation.	None
Ownership	Boolean	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 no 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.	None

Example of ownership in the business object XSD file

```
<element minOccurs="0" name="addressObj" type="bons0:OutboundRtasserAddress"
 maxOccurs="unbounded">
  <annotation>
    <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
    <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
      <jdbcasi:Ownership>true</jdbcasi:Ownership>
    </jdbcasi:JDBCAttributeTypeMetadata>
    </appinfo>
  </annotation>
</element>
```

```
<element minOccurs="0" name="custinfoObj" type="bons1:OutboundRtasserCustinfo"
maxOccurs="1">
 <annotation>
  <appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
   <jdbcasi:JDBCAttributeTypeMetadata xmlns:jdbcasi=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
    <jdbcasi:Ownership>false</jdbcasi:Ownership>
   </jdbcasi:JDBCAttributeTypeMetadata>
  </appinfo>
 </annotation>
</element>
```

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.

Another example XSD file for single- and multiple-cardinality child business objects

```
<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>
```

Application-specific information for operations

The adapter uses application-specific information at the operation level to perform operations, such as to retrieve and update information in the database. The adapter retrieves and updates database tables using SQL queries, stored procedures, or stored functions, as specified in the business objects.

If you choose to add stored procedures or stored functions to the business objects, set the operation application-specific information (ASI) as specified in Table 19.

Table 19. Operation application-specific information

Operation ASI for StoredProcedure parameters element	Set by wizard	Description
Parameters	Yes	Lists the stored procedure parameters.
PropertyName	Yes	Set to the name of the business object attribute that you select.
ResultSet	No	If the stored procedure returns a result set, set this parameter to `True` in the business object definition.

Table 19. Operation application-specific information (continued)

Operation ASI for StoredProcedure parameters element	Set by wizard	Description
ReturnValue	Yes	If the stored procedure has a return value, this parameter contains one of these values: • The string RS. This value indicates that the procedure returns a result set, which is used to create the multiple-cardinality container corresponding to this business object. • The name of a business object attribute. This value indicates that procedure returns the value that is to be assigned to that particular attribute in the business object at run time. If the attribute is another child business object, the adapter returns an error.
StoredProcedure	Yes	Set to the stored procedure name.
StoredProcedureType	Yes	You choose from a list of types. For information about valid stored procedure types, "Stored procedure type" on page 38.
Type	Yes	Set to the type of the stored procedure parameter. Possible values are: • IP (input only) • OP (output only) • IO (input and output) • RS (result set)

Business object-level application-specific information

Application-specific information in business object definitions provides the adapter with application-dependent instructions for how to process business objects. The adapter parses the application-specific information from the business object or from its attributes or operations to generate queries for Create, Update, Retrieve, and Delete operations.

Application-specific information for table and view business objects

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.

The external service wizard sets the TableName application-specific information attribute to a value in the form of *SchemaName.TableName*. It prompts you for the information necessary to perform a physical or logical Delete operation and then sets the business object-level application-specific information shown in Table 20.

Table 20. Business object application-specific information (ASI) for a table business object

Application-specific information	Type	Description
TableName	String	The name of the database table corresponding to this business object.

Table 20. Business object application-specific information (ASI) for a table business object (continued)

Application-specific information	Type	Description
StatusColumnName	String	Indicates whether the adapter is to logically or physically delete data in the table. If the StatusColumnName parameter is not set, data is physically deleted. If the parameter is set, it specifies the name of the column that indicates a logically deleted row. You specify this parameter when you select the table object in the external service wizard. This parameter applies to both Update and Delete operations.
StatusValue	String	The value that indicates that a column is logically deleted. You specify this value when you select the table object in the external service wizard.

To illustrate how the adapter determines whether to do a logical or physical delete in response to an Update or Delete operation, assume that a Customer business object has the business object application-specific information shown in Table 21.

Table 21. Sample parameters for business object application-specific information for a table business object

Application-specific information	Value
TableName	customer
StatusColumnName	status
StatusValue	deleted

Assume that the adapter receives a request to delete a customer. Because the business object includes the StatusColumnName parameter in its application-specific information, the adapter performs a logical delete operation. It does this by placing the string "deleted", which is specified in the StatusValue parameter, in the status column, which is the column specified in the StatusColumnName parameter.

Such a request causes the adapter to issue the following SQL statement:

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

However, if the StausColumnName parameter is not set, the customer records are physically deleted. The adapter issues the following SQL statement:

```
DELETE customer where pkey = . . . .
```

Application-specific information for stored procedure business objects

For business objects that are based on stored procedures, the external service wizard sets the business object-level application-specific information SPName to a value in the form of *SchemaName + SPName*. It sets the business object-level application-specific information, which is listed in Table 22 on page 175. The attributes of the business object are created based on the stored procedure input/output parameters. If the stored procedure has one returned value, a corresponding business object attribute is created. If the returned value or any of the input/output parameters are complex data types, the wizard creates child business objects for them.

The discovery of database objects in the external service wizard can support nested structures and arrays. If these child business objects are generated from returned result sets, their names are in the form of *Prefix* + *SchemaName* + *SPName* + `RetRS` + *Number*. For example, if one stored procedure returns two result sets, the wizard creates two child business objects for them. Their names will be *Prefix* + *SchemaName* + *SPName* + `RetRS1` and *Prefix* + *SchemaName* + *SPName* + `RetRS2`.

When the child business objects are generated from input/output parameters with a complex data type of ResultSet, Struct, or Array, these child business object names are in the form of *Prefix+SchemaName+SPName+ParameterName*. For those child business objects that correspond to nested structures and arrays, their business object names are in the form of *Prefix+SchemaName+SPName+ParameterName+ColumnName*.

Table 22. Business object application-specific information (ASI) for business objects based on stored procedures

Application-specific information	Type	Description
SPName	String	The name of the stored procedure or stored function
ResultSet	Boolean	Indicates whether the stored procedure or stored function will return a result set. If `true`, the stored procedure returns one or more result sets. If `false`, the stored procedure or stored function does not return a result set.
MaxNumberOfRetRS	String	The maximum number of returned result sets that are handled by the adapter runtime
ReturnValue	String	Set to the name of the corresponding business object attribute if the stored procedure has a return value. If the returned value is of simple data type, the attribute is also of simple data type. If the returned value is a result set, this attribute points to a child business object.

Application-specific information for query business objects

For query business objects, there is one business object-level application-specific information, as shown in Table 23.

Table 23. Business object application-specific information (ASI) for query business objects

Application-specific information	Type	Description
SelectStatement	String	The complete SELECT statement that performs the query. You specify the statement in the external service wizard.

The external service 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.

When the wizard generates a stored procedure business object, it creates a child business object if necessary, such as for ResultSet, Struct, and Array. Creating parent-child relationships between table business objects is done manually using the Business Object Editor.

The wizard handles business objects based on synonym/nicknames like objects based on tables and views, even when a synonym is of a stored procedure.

Application-specific information for batch SQL business objects

Batch SQL business objects have the following application-specific information,

Table 24. Business object application-specific information (ASI) for batch SQL business objects

Application-specific information	Type	Description
BatchSQLIndex	String	Specifies the order of the SQL statement. For example, if the user specifies 3 statements (separated by semicolons) in one batch SQL business object, the first statement has the index of 1, the second has the index of 2, and the last has the index of 3.

SQLStatement	String	Contains one INSERT, UPDATE, and DELETE SQL statement specified by the user. If the user specifies multiple SQL statements in one business object, each is stored in a separate SQLStatement element. For example:
		
		`Delete From Customer where pkey=?`
		
		`Insert into customer (pkey,ccode,fname,lname values(?,?,?,?)`

Application-specific information for wrapper business objects

For wrapper business objects, the Wrapper application-specific information is added and set to `True`. No other application-specific information is needed at the business object-level on a wrapper business object.

Application-specific information	Type	Description
Wrapper	Boolean	Indicates whether the business objects is a wrapper business object When Wrapper is `True`, no other business-object level ASI is needed.

Naming conventions

When the external service wizard generates a business object, it provides a name for the business object that reflects the naming convention for the adapter. Typically, the business object name indicates the structure of the business object.

When the external service wizard creates names for a business object, it replaces any special character except the underscore (_) in the business object name with `U` followed by its Unicode number. For example, the business object name for the

Order_Item table in the database is Order_Item. The business object name for the Shipping-Address table is ShippingU45Address.

Business object names have no semantic value to the adapter or the database; that is, they derive no information nor meaning from the business object name. If one name is replaced by another, the adapter behavior remains the same.

Business object names can carry database-specific metadata. A name can use a string like `JDBC` or *`%AppName%`* as a prefix to help distinguish between 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`.

For business objects that do not correspond to database objects, such as business objects for database queries, batch SQL statements, and wrappers, be careful to use names that do not duplicate names for different query business objects, otherwise the names will be overwritten.

Globalized characters are supported in any business object name.

You can rename business objects by using the refactoring functionality in WebSphere Integration Developer. For more details, refer to the WebSphere Integration Developer documentation.

The following table describes the naming conventions that the wizard uses for the business object.

Table 25. Business object naming conventions

Element	**Naming convention**
Business graph	The business graph that contains the parent business object is named for the contained business object, followed by the string "BG"; for example, the business graph that contains the SalesCustomer business object is named SalesCustomerBG.
Business objects for: • Tables • Views • Stored procedures • Stored functions • Synonyms and nicknames	For those business objects that are based on tables, views, stored procedures, and synonyms and nicknames, the external service wizard generates the name of the business object in the form of *Prefix* + *SchemaName* + *ObjectName*, where: • *Prefix* is the value as specified in the external service 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, stored function, or synonym/nickname. 2 For example, using the prefix Campaign12 for the Customer table in the Sales schema, 2 the business object name is Campaign12SalesCustomer.
Query business objects	For query business objects, the external service wizard generates the name of the business object in the form of *Prefix* + *QueryBOName*, where: • *Prefix* is the prefix you specify in the wizard. A prefix is not required, and if not specified, no prefix will be added to the business object name. • *QueryBOName* is the value you specified when you configured the business object in the wizard.

Table 25. Business object naming conventions (continued)

Element	Naming convention
BatchSQL business objects	For batchSQL business objects, the external service wizard generates the name of the business object in the form of *Prefix* + *BatchSQLBOName*, where: • *Prefix* is the prefix you specify in the wizard. A prefix is not required, and if not specified, no prefix will be added to the business object name. • *BatchSQLBOName* is the name you specified when you configured the business object in the wizard.
Wrapper business objects	For wrapper business objects, the external service wizard generates the name of the business object in the form of *Prefix* + *WrapperBOName*, where: • *Prefix* is the prefix you specify in the wizard. A prefix is not required, and if not specified, no prefix will be added to the business object name. • *WrapperBOName* is the name you specified when you configured the business object in the wizard.

Outbound configuration properties

WebSphere Adapter for JDBC has several categories of outbound connection configuration properties, which you set with the external service wizard while generating or creating objects and services. You can change the resource adapter and managed connection factory properties after you deploy the module to WebSphere Process Server using WebSphere Integration Developer or the WebSphere Process Server administrative console, but connection properties for the external service wizard cannot be changed after deployment.

Guide to information about properties

The properties used to configure WebSphere Adapter for JDBC are described in detail in tables included in each of the configuration properties topics, such as Resource adapter properties, Managed connection factory properties, and so on. To help you use these tables, information about each row you might see is explained here.

The following table explains the meaning of each row that might be displayed in the table for a configuration property.

Row	Explanation
Required	A required field (property) must have a value in order for the adapter to work. Sometimes the external service wizard provides a default value for required properties. Removing a default value from a required field on the external service wizard *will not change that default value*. When a required field contains no value at all, the external service wizard will process the field using its assigned default value, and that default value will also be displayed on the administrative console. Possible values are **Yes** and **No**. Sometimes a property is required only when another property has a specific value. When this is the case, the table will note this dependency. For example, • Yes, when the EventQueryType property is set to Dynamic • Yes, for Oracle databases
Possible values	Lists and describes the possible values that you can select for the property.

Row	Explanation
Default	The predefined value that is set by the external service wizard. When the property is required, you must either accept the default value or specify one yourself. If a property has no default value, the table will state No default value. The word `None` is an acceptable default value, and does not mean that there is no default value.
Unit of measure	Specifies how the property is measured, for example in kilobytes or seconds.
Property type	Describes the property type. Valid property types include the following: • Boolean • String • Integer
Usage	Describes usage conditions or restrictions that might apply to the property. For instance, here is how a restriction would be documented: For WebSphere Application Server version 6.40 or earlier, the password: • Must be uppercase • Must be 8 characters in length For versions of WebSphere Application Server later than 6.40, the password: • Is not case sensitive • Can be up to 40 characters in length. This section lists other properties that affect this property or that are affected by this property and describes the nature of the conditional relationship.
Example	Provides sample property values, for example: ″If Language is set to JA (Japanese), Codepage number is set to `8000`″.
Globalized	If a property is globalized, it has national language support, meaning that you can set the value in your national language. Valid values are **Yes** and **No**.
Bidi supported	Indicates whether the property is supported in bidirectional (bidi) processing. Bidirectional processing pertains to the task of processing data that contains both left-to-right (Hebrew or Arabic, for example) and right-to-left (a URL or file path, for example) semantic content within the same file. Valid values are **Yes** and **No**.

Connection properties for the wizard

External service connection properties are used to establish a connection between the external service wizard, a tool that is used to create business objects, and the database. These properties specify such things as connection configuration, bidirectional transformation properties, and logging options for the wizard. Once a connection is established, the wizard can discover in the database the metadata it needs to create business objects.

Some of the properties that you provide for the wizard to use to discover objects in the database are used as the initial value for the runtime properties that you specify later in the wizard. These include resource adapter, managed connection factory, and activation specification properties.

The connection properties for the external service wizard and their purpose are described in the following table. A complete description of each property is provided in the sections that follow the table. For information about how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 178.

Table 26. Connection properties for the external service wizard

Property name in the wizard	Description
Additional JDBC driver connection properties	Additional properties for connecting to the database using the JDBC driver, which are used in addition to the UserName and Password properties
"Database"	The database name
"Database software" on page 181	The name and version of the database management software that the adapter will access
Database URL	The database URL that is used to connect to the database
"Host name" on page 182	The host name or IP address of the database server
"JDBC driver classname" on page 182	The name of the JDBC driver class
"JDBC driver type" on page 183	The type of JDBC driver to use
Password	Password for the corresponding user name.
"Port number" on page 183	The port number for connecting to the database instance
"Prefix for business object names" on page 183	A prefix to be added to the name of the business object
User name	The database user name

The external service wizard uses the bidirectional connection properties to apply the proper bidirectional transformation on the data passed to the enterprise information system.

Additional JDBC driver connection properties

This property contains additional information for connecting to the database using the JDBC driver.

Table 27. Driver connection properties details

Required	No
Possible values	Database connection properties are database-specific.
Default	No default value
Property type	String
Usage	These connection properties are used in addition to the UserName and Password properties. They allow you to customize the database connection used by the adapter. Specify the connection properties as one or more *name*:*value* pairs, separated by the semicolon character (;).
Example	The following value of this property specifies a login time out interval, makes a read-only database connection, and sets the security mechanism: `loginTimeout:20;readOnly:true;securityMechanism:USER_ONLY_SECURITY`
Globalized	Yes
Bidi supported	No

Database

This property specifies the name of the database.

Table 28. Database details

Required	Yes
Default	The default value is database-specific
Property type	String
Usage	This is the name of the database you want to access. For an Oracle database, this is the system ID (SID), which identifies the database.
Globalized	Yes
Bidi supported	Yes

Database software

This property specifies the database management software that manages the database that the adapter will access.

Table 29. Database software details

Row	**Explanation**
Required	Yes
Possible values	This property lists common database software by name and version number. If your software is not listed, select `Generic JDBC`.
Default	No default value
Property type	String
Usage	The external service wizard uses the value of this property to set default values and generate database-specific selection lists for other properties. For example, if you select `DB2 USB Version 9.1`, the JDBC driver class field in the wizard displays only the JDBC drivers supported by that version of DB2 UDB. If you select `Oracle 10`, a different set of JDBC drivers is shown.
Globalized	Yes
Bidi supported	Yes

Database URL

This property specifies the JDBC driver-specific URL for creating a connection to the database.

Table 30. Database URL details

Required	Yes, for specific JDBC drivers
Default	No default value
Property type	String
Usage	This value is specific to the database software and JDBC driver that you are using. If your database server supports IPv6, you can specify the host name portion of the database URL in IPv6 format. Enclose the IP address in square brackets (`[]`).

Table 30. Database URL details (continued)

Examples	The following are typical values for common database servers.
	DB2 universal (type 4) JDBC driver
	`jdbc:db2://www.example.com:50000/DB`
	DB2 universal JDBC driver with an IPv6 address
	`jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB`
	DB2 Universal Database type driver for local connection
	`jdbc:db2:TEST`
	DB2 Universal Database type 2 driver for remote connection
	`jdbc:db2://www.example.com:50000/TEST`
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`

Globalized	Yes
Bidi supported	Yes

Host name

This property specifies the host name or IP address of the database server.

Table 31. Host name details

Row	**Explanation**
Required	Yes
Default	No default value
Property type	String
Usage	This is the host name or IP address of the database server. If the database server supports IPv6, you can specify the host name in IPv6 format.
Globalized	Yes
Bidi supported	Yes

JDBC driver classname

This property specifies the name of the JDBC driver class.

Table 32. JDBC driver classname details

Row	**Explanation**
Required	Yes
Possible values	The possible values depend on the database type and version. The wizard displays a list of known drivers.
Default	The default depends on the database type and version.
Property type	String
Usage	This is the class name of JDBC driver. The wizard displays the default class name for the JDBC driver type you select, but you can type another class name if needed. The class name must be in the JDBC driver files you provide when you start the wizard.
Globalized	Yes
Bidi supported	No

JDBC driver type

This property specifies the type of JDBC driver to use.

Table 33. JDBC driver type details

Row	Explanation
Required	Yes
Possible values	The possible values depend on the database type and version. The wizard displays a list of known drivers.
Default	The default depends on the database type and version
Property type	String
Usage	This is the type of JDBC driver to use. While the basic question is whether you want a type 2 or type 4 (universal) driver, each database system has its own name for the driver. The wizard displays a list of drivers known for each database system, Select `Other` if your driver is not listed. The information in this field must agree with the JDBC driver files you provide when you start the wizard.
Globalized	Yes
Bidi supported	No

Password (Password)

This property specifies the password for the database user name.

Table 34. Password details

Required	Yes, unless the DataSourceJNDIName or XADataSourceName properties are set
Default	No default value
Property type	String
2 Usage	If this property is set, it overrides the password specified on a data source on the server using either the DataSourceJNDIName or XADataSourceName properties.
Globalized	Yes
Bidi supported	Yes

Port number

This property specifies the port number for the database instance.

Table 35. Port number details

Required	Yes
Default	The default value is database-specific, and is initialized by the wizard.
Property type	String
Usage	This is the port number for connecting to the database instance.
Globalized	Yes
Bidi supported	No

Prefix for business object names

The prefix to be added to the name of the business object.

Table 36. Prefix details

Required	No
Default	No default value
Property type	String
Usage	Use a prefix to help distinguish between the types of business objects.
Example	You might specify a prefix of `JDBC` for generic business objects and `%AppName%` for an application-specific business object.
Globalized	Yes
Bidi supported	No

User name (UserName)

This property specifies the database user name for connecting to the database.

Table 37. User name details

Required	Yes, unless the DataSourceJNDIName or XADataSourceName properties are set
Default	No default value
Property type	String
2 Usage	If this property is set, it overrides the user name specified on a data source on the server using either the DataSourceJNDIName or XADataSourceName properties.
Globalized	Yes
Bidi supported	Yes

Resource adapter properties

The resource adapter properties control the general operation of the adapter, such as specifying the namespace for business objects. You set the resource adapter properties using the external service wizard when you configure the adapter. After deploying the adapter, use the administrative console to change these properties.

The following properties for logging and tracing are no longer required in version 6.1.0, but are supported for compatibility with previous versions.
- LogFileMaxSize
- LogFileName
- LogNumberOfFiles
- TraceFileMaxSize
- TraceFileName
- TraceNumberOfFiles

The following properties that were specified for both inbound and outbound processing in version 6.0.2 apply only to inbound processing in version 6.1.0. For outbound processing, these properties are now located in the managed connection factory property group:
- PingQuery
- QueryTimeOut
- ReturnDummyBOForSP

The BusinessObjectNameSpace property has moved to the activation specification properties.

The following table lists the resource adapter properties and their purpose. A complete description of each property is provided in the sections that follow the table.

Table 38. Resource adapter properties for the Adapter for JDBC

Name		
In the wizard	**In the administrative console**	**Description**
Adapter ID to use for logging and tracing	AdapterID	Identifies the adapter instance for CEI and PMI events with respect to logging and tracing.
Database vendor	DatabaseVendor	The type of database that the adapter uses for special processing
Query time out	QueryTimeOut	The maximum number of seconds a query can take for all SQL statements
Return business object even when the stored procedure result set is empty	ReturnDummyBOForSP	Specifies whether to return output parameters when the result set is empty
(Not available)	enableHASupport	Do not change this property.
(Not available)	LogFileMaxSize	Supported for compatibility with earlier versions
(Not available)	LogFilename	Supported for compatibility with earlier versions
(Not available)	LogNumberOfFiles	Supported for compatibility with earlier versions
SQL query to verify the connection	PingQuery	The SQL query used to test valid connection to the database
(Not available)	TraceFileMaxSize	Supported for compatibility with earlier versions
(Not available)	TraceFileName	Supported for compatibility with earlier versions
(Not available)	TraceNumberOfFiles	Supported for compatibility with earlier versions

Adapter ID to use for logging and tracing (AdapterID)

Use this property to identify a specific deployment, or instance, of the adapter.

Table 39. Adapter ID to use for logging and tracing details

Required	Yes
Default	CWYBC_JDBC
Property type	String
Usage	This property is used to identify the adapter instance for PMI events. If you are deploying multiple instances of an adapter, set this property to a unique value for each adapter instance. For inbound processing this property is retrieved from the resource adapter properties. For outbound processing, it is retrieved form the managed connection factory properties.
Globalized	Yes
Bidi supported	No

Database vendor (DatabaseVendor)

This property specifies which type of database is used. The type is determined by the database vendor name.

Table 40. Database vendor details

Required	Yes

Possible values	`DB2`
	`MSSQLServer`
	`Oracle`
	`Others`

Default	No default value
Property type	String
Usage	Some SQL statements require special processing, which varied by database type. For example, the Struct and Array data types in Oracle require special processing. This property specifies which RDBMS is used, which determines the database type. Specify the value that corresponds to your database vendor, as follows: • `DB2`, for an IBM DB2 database • `Oracle`, for an Oracle database • `MSSQLServer`, for a Microsoft SQL Server database • `Others`, for all other database types For other database types, the adapter does not perform any special processing. Make sure that the correct driver is specified in the JDBCDriverClass property.
Globalized	No
Bidi supported	No

Enable high availability support (enableHASupport)

Do not change this property. It must be set to `true`.

Query timeout (QueryTimeOut)

This property specifies the maximum number of seconds a query can take for all SQL statements.

Table 41. Query timeout details

Required	No
Default	No default value
Unit of measure	Seconds
Property type	Integer
Usage	If the query takes longer than the number of seconds specified, the database generates an SQL exception that is captured. The associated message is logged in the log file. If a value is not specified, no timeout is set on the query.
Globalized	Yes
Bidi supported	No

Return business object even when the stored procedure result set is empty (ReturnDummyBOForSP)

This property specifies whether to return output parameters when the result set is empty.

Table 42. Return business object even when the stored procedure result set is empty details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	The Retrieve Stored Procedure (RetrieveSP) operation returns a result set. If the result set is empty and the ReturnDummyBOForSP property is set to `False`, no business objects are created, and the output parameters returned by the procedure call cannot be retrieved. However, if ReturnDummyBOForSP is `True`, a dummy business object is created and populated with values from output and input/output parameters in the corresponding attributes.
Globalized	Yes
Bidi supported	No

SQL query to verify the connection (PingQuery)

This property specifies the SQL query that is used to test valid connection to the database.

Table 43. Ping query details

Required	No
Property type	String
Default	No default value
Usage	This property contains the SQL query statement that you want to run to determine whether the adapter can connect to the database.
Globalized	No
Bidi supported	No

Managed connection factory properties

Managed connection factory properties are used by the adapter at run time to create an outbound connection instance with the database.

You set managed connection factory properties using the external service wizard during adapter configuration. You can change them using the WebSphere Integration Developer assembly editor or, after deployment, with the WebSphere Process Server or WebSphere Enterprise Service Bus administrative console.

The following table lists and describes the managed connection factory properties. A complete description of each property is provided in the sections that follow the table. For information about how to read the property detail tables in the sections that follow, see “Guide to information about properties” on page 178.

Note: The external service wizard refers to these properties as managed connection factory properties, while the administrative console refers to them as J2C connection factory properties.

Table 44. Managed connection factory properties for Adapter for JDBC

Property name		
In the wizard	**In the administrative console**	**Description**
Adapter ID	AdapterID	Identifies the adapter instance for CEI and PMI events with respect to logging and tracing.
Additional JDBC driver connection properties	DriverConnectionProperties	Additional properties for connecting to the database using the JDBC driver, which are used in addition to the UserName and Password properties
Auto commit	AutoCommit	The AutoCommit value to use on the connection
Data source JNDI name	DataSourceJNDIName	The name of the JNDI data source to use to establish a connection to the database
Database URL	DatabaseURL	The database URL that is used to connect to the database
Database vendor	DatabaseVendor	The type of database that the adapter uses for special processing
JDBC driver class	JDBCDriverClass	The class name of the JDBC driver that is used to connect to the database
Password	Password	Password for the corresponding user name.
Query timeout	QueryTimeOut	The maximum number of seconds a query can take for all SQL statements
Return business object even when the stored procedure result set is empty	ReturnDummyBOForSP	Specifies whether to return output parameters when the result set is empty
SQL query to connection	PingQuery	The SQL query used to test valid connection to the database
User name	UserName	The database user name
XA data source name	XADataSourceName	The name of the XA data source to use to establish an connection to the database for XA (distributed) transactions
XA database name	XADatabaseName	The database name used for the XA connection

Adapter ID to use for logging and tracing (AdapterID)

Use this property to identify a specific deployment, or instance, of the adapter.

Table 45. Adapter ID to use for logging and tracing details

Required	Yes
Default	CWYBC_JDBC
Property type	String
Usage	This property is used to identify the adapter instance for PMI events. If you are deploying multiple instances of an adapter, set this property to a unique value for each adapter instance. For inbound processing this property is retrieved from the resource adapter properties. For outbound processing, it is retrieved form the managed connection factory properties.
Globalized	Yes
Bidi supported	No

Additional JDBC driver connection properties (DriverConnectionProperties)

This property contains additional information for connecting to the database using the JDBC driver.

Table 46. Driver connection properties details

Required	No
Possible values	Database connection properties are database-specific.
Default	No default value
Property type	String
Usage	These connection properties are used in addition to the UserName and Password properties. They allow you to customize the database connection used by the adapter. Specify the connection properties as one or more *name*:*value* pairs, separated by the semicolon character (;).
Example	The following value of this property specifies a login time out interval, makes a read-only database connection, and sets the security mechanism: `loginTimeout:20;readOnly:true;securityMechanism:USER_ONLY_SECURITY`
Globalized	Yes
Bidi supported	No

Auto commit (AutoCommit)

This property specifies whether AutoCommit is set for the connection.

Table 47. Auto commit details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	This property is ignored for XA (distributed) transactions.
Globalized	No
Bidi supported	No

Data source JNDI name (DataSourceJNDIName)

This property specifies the name of the JNDI data source to use to establish a connection to the database.

Table 48. Data source JNDI name details

Required	No
Default	No default value
Property type	String

Table 48. Data source JNDI name details (continued)

Usage	2 Use this property to specify the JNDI name of a data source in WebSphere Process Server or 2 WebSphere Enterprise Service Bus that specifies connection information for the target database. To improve the performance of inbound or outbound operations, specify the name of a data source that is enabled for prepared statement caching. If the user name and password properties are also set, they override the user name and password in the data source. The properties for connecting to the database are used in the following order: 1. If the DataSourceJNDIName property is set, the adapter uses it to establish the connection to the database. If the UserName and Password properties are also set, they override the user name and password set on the data source. 2. If the DataSourceJNDIName property is not set, the adapter uses the XADataSourceName and XADatabaseName properties, if set, to establish the connection. 2 The DataSourceJNDIName property represents an XA or connection pool data source. If you 2 define a JNDI data source on the server that supports XA transactions and then specify that 2 data source when you configure the adapter, you can connect to any type of database that 2 supports XA transactions. If you use an XA data source and database, the adapter supports 2 XA transaction only for DB2 and Oracle databases. 3. If the DataSourceJNDIName, XADataSourceName, and XADatabaseName properties are not 2 set, then the adapter uses the DatabaseURL, JDBCDriverClass, UserName and Password properties to establish the connection. Do not confuse the data source JNDI name property with the JNDI name of a managed connection factory or activation specification on the server. The following list highlights important differences between the types of JNDI names: • Data source JNDI name – Specifies a connection to a database – Used instead of saving user name and password in adapter properties – Saved as an adapter property • JNDI name of the managed connection factory or activation specification – Specifies a connection to a managed connection factory or activation specification on the server – Used instead of specifying the value of each managed connection factory or activation specification property in the wizard – Saved as the connection target in the import file
Globalized	Yes
Bidi supported	No

Database URL (DatabaseURL)

This property specifies the JDBC driver-specific URL for creating a connection to the database.

Table 49. Database URL details

Required	Yes, unless one of the following sets of properties are set: • The DataSourceJNDIName property • The XADataSourceName and XADatabaseName properties
Default	No default value

Table 49. Database URL details (continued)

Property type	String
Usage	In the external service wizard, you compose the database URL by filling in database-specific fields. For example, the database URL for a DB2 database is composed of the database name, the server host name, and the database port number. In the administrative console, type the complete database URL value. If your database server supports IPv6, you can specify the host name portion of the database URL in IPv6 format. The properties for connecting to the database are used in the following order: 1. If the DataSourceJNDIName property is set, the adapter uses it to establish the connection to the database. If the UserName and Password properties are also set, they override the user name and password set on the data source. 2. If the DataSourceJNDIName property is not set, the adapter uses the XADataSourceName and XADatabaseName properties, if set, to establish the connection. The DataSourceJNDIName property represents an XA or connection pool data source. If you define a JNDI data source on the server that supports XA transactions and then specify that data source when you configure the adapter, you can connect to any type of database that supports XA transactions. If you use an XA data source and database, the adapter supports XA transaction only for DB2 and Oracle databases. 3. If the DataSourceJNDIName, XADataSourceName, and XADatabaseName properties are not set, then the adapter uses the DatabaseURL, JDBCDriverClass, UserName and Password properties to establish the connection. If you specify the host name as an IP address in IPv6 format, enclose the IP address in square brackets ([]).

Examples	The following are typical values for common database servers:
	
	DB2 universal (type 4) JDBC driver
	`jdbc:db2://www.example.com:50000/DB`
	
	DB2 universal JDBC driver with an IPv6 address
	`jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB`
	
	DB2 Universal Database type 2 driver for local connection
	jdbc:db2:TEST
	
	DB2 Universal Database type 2 driver for remote connection
	jdbc:db2://www.example.com:50000/TEST
	
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`

Globalized	Yes
Bidi supported	Yes

Database vendor (DatabaseVendor)

This property specifies which type of database is used. The type is determined by the database vendor name.

Table 50. Database vendor details

Required	Yes

Possible values	`DB2`
	`MSSQLServer`
	`Oracle`
	`Others`

Table 50. Database vendor details (continued)

Default	No default value
Property type	String
Usage	Some SQL statements require special processing, which varied by database type. For example, the Struct and Array data types in Oracle require special processing. This property specifies which RDBMS is used, which determines the database type. Specify the value that corresponds to your database vendor, as follows: • `DB2`, for an IBM DB2 database • `Oracle`, for an Oracle database • `MSSQLServer`, for a Microsoft SQL Server database • `Others`, for all other database types For other database types, the adapter does not perform any special processing. Make sure that the correct driver is specified in the JDBCDriverClass property.
Globalized	No
Bidi supported	No

JDBC driver class (JDBCDriverClass)

This property specifies the class name of the JDBC driver that is used to connect to the database.

Table 51. JDBC driver class details

Required	Yes, if the DataSourceJNDIName property is not set
Possible values	Values are database-specific.
Default	No default value
Property type	String
Usage	In the external service wizard, the driver is specified for you if you select a common database software and driver combination, such as type 4 drivers for recent versions of IBM DB2, Oracle, and Microsoft SQL. For most type 2 drivers for most database software, you must provide the database class name. If you select a type 2 driver or a generic driver, you must type the JDBC driver class name. For example, for the DB2 Universal Database type 2 driver, the class name is `COM.ibm.db2.jdbc.app.DB2Driver` In the administrative console, type the database-specific name of the driver. If the DataSourceJNDIName property is set, this property is ignored.

Table 51. JDBC driver class details (continued)

Examples	**In the external service wizard:**
	• To connect to a DB2 database using the universal, or type 4, JDBC driver, select`IBM DB2 Universal`.
	• To connect to a DB2 database using the DB2 universal type 2 driver, select `Other`.
	• To connect to an Oracle 10 database using the type 4 driver, select `Oracle Thin Driver`.
	
	In the administrative console:
	
	DB2 Universal Database type 2 driver
	`COM.ibm.db2.jdbc.app.DB2Driver`
	
	DB2 Universal Database type 4 driver
	`com.ibm.db2.jcc.DB2Driver`
	
	Oracle Thin JDBC driver
	`oracle.jdbc.driver.OracleDriver`
	
	IBM Toolkit for Java remote driver for i5/OS
	`com.ibm.as400.access.AS400JDBCDriver`
	
	IBM WebSphere Connect JDBC driver for Microsoft SQL Server
	`com.ibm.websphere.jdbc.sqlserver.SQLServerDriver`

Globalized	No
Bidi supported	No

Password (Password)

This property specifies the password for the database user name.

Table 52. Password details

Required	Yes, unless the DataSourceJNDIName or XADataSourceName properties are set
Default	No default value
Property type	String
Usage	If this property is set, it overrides the password specified on a data source on the server using either the DataSourceJNDIName or XADataSourceName properties.
Globalized	Yes
Bidi supported	Yes

2

Query timeout (QueryTimeOut)

This property specifies the maximum number of seconds a query can take for all SQL statements.

Table 53. Query timeout details

Required	No
Default	No default value
Unit of measure	Seconds
Property type	Integer
Usage	If the query takes longer than the number of seconds specified, the database generates an SQL exception that is captured. The associated message is logged in the log file. If a value is not specified, no timeout is set on the query.

Table 53. Query timeout details (continued)

Globalized	Yes
Bidi supported	No

Return business object even when the stored procedure result set is empty (ReturnDummyBOForSP)

This property specifies whether to return output parameters when the result set is empty.

Table 54. Return business object even when the stored procedure result set is empty details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	The Retrieve Stored Procedure (RetrieveSP) operation returns a result set. If the result set is empty and the ReturnDummyBOForSP property is set to `False`, no business objects are created, and the output parameters returned by the procedure call cannot be retrieved. However, if ReturnDummyBOForSP is `True`, a dummy business object is created and populated with values from output and input/output parameters in the corresponding attributes.
Globalized	Yes
Bidi supported	No

SQL query to verify the connection (PingQuery)

This property specifies the SQL query that is used to test valid connection to the database.

Table 55. Ping query details

Required	No
Property type	String
Default	No default value
Usage	This property contains the SQL query statement that you want to run to determine whether the adapter can connect to the database.
Globalized	No
Bidi supported	No

User name (UserName)

This property specifies the database user name for connecting to the database.

Table 56. User name details

Required	Yes, unless the DataSourceJNDIName or XADataSourceName properties are set
Default	No default value
Property type	String
Usage	If this property is set, it overrides the user name specified on a data source on the server using either the DataSourceJNDIName or XADataSourceName properties.

2

Table 56. User name details (continued)

Globalized	Yes
Bidi supported	Yes

XA data source name (XADataSourceName)

This property specifies the name of the XA data source to use to establish a connection to the database for XA (distributed) transactions.

Table 57. XA data source name details

Required	No
Default	No default value
Property type	String
Usage	To make an XA connection for a DB2 database, this property is used in conjunction with XADatabaseName property. To make an XA connection to an Oracle database, this property is used but the XADatabaseName property is not. If the DataSourceJNDIName property is specified, this property is ignored.

Examples	A typical value for a DB2 database with a type 2 JDBC driver (db2java.zip):
	
	`COM.ibm.db2.jdbc.DB2XADataSource`
	
	A typical value for a DB2 database with a type 4 JDBC driver (db2jcc.jar):
	
	`com.ibm.db2.jcc.DB2XADataSource`
	
	A typical value for an Oracle database:
	
	`oracle.jdbc.xa.client.OracleXADataSource`

Globalized	No
Bidi supported	No

XA database name (XADatabaseName)

This property specifies the name of the database to use for the XA connection.

Table 58. XA database name details

Required	No
Default	No default value
Property type	String
Usage	To make an XA connection to a DB2 database, this property is used in conjunction with the XADataSourceName property. This property is not required for Oracle databases. If the DataSourceJNDIName property is specified, this property is ignored.
Globalized	Yes
Bidi supported	Yes

Interaction specification properties

Interaction specification, or InteractionSpec, properties control the interaction for an operation. The external service wizard sets the interaction specification properties when you configure the adapter. Typically, you do not need to change these properties. However, some properties for outbound operations can be changed by the user. For example, you might increase the value of the interaction specification property that specifies the maximum number of records to be returned by a RetrieveAll operation, if your RetrieveAll operations do not return complete information. To change these properties after the application is deployed, use the assembly editor in WebSphere Integration Developer. The properties reside in the method binding of the import.

Table 59 lists and describes the interaction specification property that you set. For information about how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 178.

Table 59. Interaction specification property for the Adapter for JDBC

Property name	Description
"Maximum records for RetrieveAll operation"	Maximum number of result sets to return during a RetrieveAll operation

Maximum records for RetrieveAll operation

This property specifies the maximum number of records to return for a RetrieveAll operation.

Table 60. Maximum records for RetrieveAll operation details

Required	Yes
Default	100
Usage	If the number of matches in the database exceeds the value of this property, the adapter throws the MatchesExceededLimitException exception and MatchesExceededLimitFault fault. If a RetrieveAll operation does not return all of the records, increase the value. If you experience out-of-memory issues, reduce the value.
Property type	Integer
Globalized	No
Bidi supported	No

Inbound configuration properties

WebSphere Adapter for JDBC has several categories of inbound connection configuration properties, which you set with the external service wizard while generating or creating objects and services. You can change the resource adapter and activation specification properties after you deploy the module using WebSphere Integration Developer or the WebSphere Process Server administrative console, but connection properties for the external service wizard cannot be changed after deployment.

Guide to information about properties

The properties used to configure WebSphere Adapter for JDBC are described in detail in tables included in each of the configuration properties topics, such as Resource adapter properties, Managed connection factory properties, and so on. To help you use these tables, information about each row you might see is explained here.

The following table explains the meaning of each row that might be displayed in the table for a configuration property.

Row	**Explanation**
Required	A required field (property) must have a value in order for the adapter to work. Sometimes the external service wizard provides a default value for required properties. Removing a default value from a required field on the external service wizard *will not change that default value*. When a required field contains no value at all, the external service wizard will process the field using its assigned default value, and that default value will also be displayed on the administrative console. Possible values are **Yes** and **No**. Sometimes a property is required only when another property has a specific value. When this is the case, the table will note this dependency. For example, • Yes, when the EventQueryType property is set to Dynamic • Yes, for Oracle databases
Possible values	Lists and describes the possible values that you can select for the property.
Default	The predefined value that is set by the external service wizard. When the property is required, you must either accept the default value or specify one yourself. If a property has no default value, the table will state No default value. The word `None` is an acceptable default value, and does not mean that there is no default value.
Unit of measure	Specifies how the property is measured, for example in kilobytes or seconds.
Property type	Describes the property type. Valid property types include the following: • Boolean • String • Integer
Usage	Describes usage conditions or restrictions that might apply to the property. For instance, here is how a restriction would be documented: For WebSphere Application Server version 6.40 or earlier, the password: • Must be uppercase • Must be 8 characters in length For versions of WebSphere Application Server later than 6.40, the password: • Is not case sensitive • Can be up to 40 characters in length. This section lists other properties that affect this property or that are affected by this property and describes the nature of the conditional relationship.
Example	Provides sample property values, for example: ″If Language is set to JA (Japanese), Codepage number is set to `8000`″.
Globalized	If a property is globalized, it has national language support, meaning that you can set the value in your national language. Valid values are **Yes** and **No**.
Bidi supported	Indicates whether the property is supported in bidirectional (bidi) processing. Bidirectional processing pertains to the task of processing data that contains both left-to-right (Hebrew or Arabic, for example) and right-to-left (a URL or file path, for example) semantic content within the same file. Valid values are **Yes** and **No**.

Connection properties for the wizard

External service connection properties are used to establish a connection between the external service wizard, a tool that is used to create business objects, and the database. These properties specify such things as connection configuration, bidirectional transformation properties, and logging options for the wizard. Once a connection is established, the wizard can discover in the database the metadata it needs to create business objects.

Some of the properties that you provide for the wizard to use to discover objects in the database are used as the initial value for the runtime properties that you specify later in the wizard. These include resource adapter, managed connection factory, and activation specification properties.

The connection properties for the external service wizard and their purpose are described in the following table. A complete description of each property is provided in the sections that follow the table. For information about how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 178.

Table 61. Connection properties for the external service wizard

Property name in the wizard	Description
Additional JDBC driver connection properties	Additional properties for connecting to the database using the JDBC driver, which are used in addition to the UserName and Password properties
"Database" on page 199	The database name
"Database software" on page 199	The name and version of the database management software that the adapter will access
Database URL	The database URL that is used to connect to the database
"Host name" on page 200	The host name or IP address of the database server
"JDBC driver classname" on page 200	The name of the JDBC driver class
"JDBC driver type" on page 201	The type of JDBC driver to use
Password	Password for the corresponding user name.
"Port number" on page 201	The port number for connecting to the database instance
"Prefix for business object names" on page 202	A prefix to be added to the name of the business object
User name	The database user name

The external service wizard uses the bidirectional connection properties to apply the proper bidirectional transformation on the data passed to the enterprise information system.

Additional JDBC driver connection properties

This property contains additional information for connecting to the database using the JDBC driver.

Table 62. Driver connection properties details

Required	No
Possible values	Database connection properties are database-specific.
Default	No default value
Property type	String

Table 62. Driver connection properties details (continued)

Usage	These connection properties are used in addition to the UserName and Password properties. They allow you to customize the database connection used by the adapter. Specify the connection properties as one or more *name*:*value* pairs, separated by the semicolon character (;).
Example	The following value of this property specifies a login time out interval, makes a read-only database connection, and sets the security mechanism: `loginTimeout:20;readOnly:true;securityMechanism:USER_ONLY_SECURITY`
Globalized	Yes
Bidi supported	No

Database

This property specifies the name of the database.

Table 63. Database details

Required	Yes
Default	The default value is database-specific
Property type	String
Usage	This is the name of the database you want to access. For an Oracle database, this is the system ID (SID), which identifies the database.
Globalized	Yes
Bidi supported	Yes

Database software

This property specifies the database management software that manages the database that the adapter will access.

Table 64. Database software details

Row	**Explanation**
Required	Yes
Possible values	This property lists common database software by name and version number. If your software is not listed, select `Generic JDBC`.
Default	No default value
Property type	String
Usage	The external service wizard uses the value of this property to set default values and generate database-specific selection lists for other properties. For example, if you select `DB2 USB Version 9.1`, the JDBC driver class field in the wizard displays only the JDBC drivers supported by that version of DB2 UDB. If you select `Oracle 10`, a different set of JDBC drivers is shown.
Globalized	Yes
Bidi supported	Yes

Database URL

This property specifies the JDBC driver-specific URL for creating a connection to the database.

Table 65. Database URL details

Required	Yes, for specific JDBC drivers
Default	No default value
Property type	String
Usage	This value is specific to the database software and JDBC driver that you are using. If your database server supports IPv6, you can specify the host name portion of the database URL in IPv6 format. Enclose the IP address in square brackets ([]).

Examples	The following are typical values for common database servers.
	
	DB2 universal (type 4) JDBC driver
	`jdbc:db2://www.example.com:50000/DB`
	
	DB2 universal JDBC driver with an IPv6 address
	`jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB`
	
	DB2 Universal Database type driver for local connection
	`jdbc:db2:TEST`
	
	DB2 Universal Database type 2 driver for remote connection
	`jdbc:db2://www.example.com:50000/TEST`
	
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`

Globalized	Yes
Bidi supported	Yes

Host name

This property specifies the host name or IP address of the database server.

Table 66. Host name details

Row	**Explanation**
Required	Yes
Default	No default value
Property type	String
Usage	This is the host name or IP address of the database server. If the database server supports IPv6, you can specify the host name in IPv6 format.
Globalized	Yes
Bidi supported	Yes

JDBC driver classname

This property specifies the name of the JDBC driver class.

Table 67. JDBC driver classname details

Row	**Explanation**
Required	Yes
Possible values	The possible values depend on the database type and version. The wizard displays a list of known drivers.
Default	The default depends on the database type and version.
Property type	String

Table 67. JDBC driver classname details (continued)

Row	Explanation
Usage	This is the class name of JDBC driver. The wizard displays the default class name for the JDBC driver type you select, but you can type another class name if needed. The class name must be in the JDBC driver files you provide when you start the wizard.
Globalized	Yes
Bidi supported	No

JDBC driver type

This property specifies the type of JDBC driver to use.

Table 68. JDBC driver type details

Row	Explanation
Required	Yes
Possible values	The possible values depend on the database type and version. The wizard displays a list of known drivers.
Default	The default depends on the database type and version
Property type	String
Usage	This is the type of JDBC driver to use. While the basic question is whether you want a type 2 or type 4 (universal) driver, each database system has its own name for the driver. The wizard displays a list of drivers known for each database system, Select `Other` if your driver is not listed. The information in this field must agree with the JDBC driver files you provide when you start the wizard.
Globalized	Yes
Bidi supported	No

Password (Password)

This property specifies the password for the database user name.

Table 69. Password details

Required	Yes, unless the DataSourceJNDIName or XADataSourceName properties are set
Default	No default value
Property type	String
Usage	If this property is set, it overrides the password specified on a data source on the server using either the DataSourceJNDIName or XADataSourceName properties.
Globalized	Yes
Bidi supported	Yes

2

Port number

This property specifies the port number for the database instance.

Table 70. Port number details

Required	Yes
Default	The default value is database-specific, and is initialized by the wizard.
Property type	String

Table 70. Port number details (continued)

Usage	This is the port number for connecting to the database instance.
Globalized	Yes
Bidi supported	No

Prefix for business object names

The prefix to be added to the name of the business object.

Table 71. Prefix details

Required	No
Default	No default value
Property type	String
Usage	Use a prefix to help distinguish between the types of business objects.
Example	You might specify a prefix of `JDBC` for generic business objects and `%AppName%` for an application-specific business object.
Globalized	Yes
Bidi supported	No

User name (UserName)

This property specifies the database user name for connecting to the database.

Table 72. User name details

Required	Yes, unless the DataSourceJNDIName or XADataSourceName properties are set
Default	No default value
Property type	String
Usage	2 If this property is set, it overrides the user name specified on a data source on the server using either the DataSourceJNDIName or XADataSourceName properties.
Globalized	Yes
Bidi supported	Yes

Resource adapter properties

The resource adapter properties control the general operation of the adapter, such as specifying the namespace for business objects. You set the resource adapter properties using the external service wizard when you configure the adapter. After deploying the adapter, use the administrative console to change these properties.

The following properties for logging and tracing are no longer required in version 6.1.0, but are supported for compatibility with previous versions.

- LogFileMaxSize
- LogFileName
- LogNumberOfFiles
- TraceFileMaxSize
- TraceFileName
- TraceNumberOfFiles

The following properties that were specified for both inbound and outbound processing in version 6.0.2 apply only to inbound processing in version 6.1.0. For outbound processing, these properties are now located in the managed connection factory property group:

- PingQuery
- QueryTimeOut
- ReturnDummyBOForSP

The BusinessObjectNameSpace property has moved to the activation specification properties.

The following table lists the resource adapter properties and their purpose. A complete description of each property is provided in the sections that follow the table.

Table 73. Resource adapter properties for the Adapter for JDBC

Name		
In the wizard	**In the administrative console**	**Description**
Adapter ID to use for logging and tracing	AdapterID	Identifies the adapter instance for CEI and PMI events with respect to logging and tracing.
Database vendor	DatabaseVendor	The type of database that the adapter uses for special processing
Query time out	QueryTimeOut	The maximum number of seconds a query can take for all SQL statements
Return business object even when the stored procedure result set is empty	ReturnDummyBOForSP	Specifies whether to return output parameters when the result set is empty
(Not available)	enableHASupport	Do not change this property.
(Not available)	LogFileMaxSize	Supported for compatibility with earlier versions
(Not available)	LogFilename	Supported for compatibility with earlier versions
(Not available)	LogNumberOfFiles	Supported for compatibility with earlier versions
SQL query to verify the connection	PingQuery	The SQL query used to test valid connection to the database
(Not available)	TraceFileMaxSize	Supported for compatibility with earlier versions
(Not available)	TraceFileName	Supported for compatibility with earlier versions
(Not available)	TraceNumberOfFiles	Supported for compatibility with earlier versions

Adapter ID to use for logging and tracing (AdapterID)

Use this property to identify a specific deployment, or instance, of the adapter.

Table 74. Adapter ID to use for logging and tracing details

Required	Yes
Default	CWYBC_JDBC
Property type	String

Table 74. Adapter ID to use for logging and tracing details (continued)

Usage	This property is used to identify the adapter instance for PMI events. If you are deploying multiple instances of an adapter, set this property to a unique value for each adapter instance. For inbound processing this property is retrieved from the resource adapter properties. For outbound processing, it is retrieved form the managed connection factory properties.
Globalized	Yes
Bidi supported	No

Database vendor (DatabaseVendor)

This property specifies which type of database is used. The type is determined by the database vendor name.

Table 75. Database vendor details

Required	Yes

Possible values	`DB2`
	`MSSQLServer`
	`Oracle`
	`Others`

Default	No default value
Property type	String
Usage	Some SQL statements require special processing, which varied by database type. For example, the Struct and Array data types in Oracle require special processing. This property specifies which RDBMS is used, which determines the database type. Specify the value that corresponds to your database vendor, as follows: • `DB2`, for an IBM DB2 database • `Oracle`, for an Oracle database • `MSSQLServer`, for a Microsoft SQL Server database • `Others`, for all other database types For other database types, the adapter does not perform any special processing. Make sure that the correct driver is specified in the JDBCDriverClass property.
Globalized	No
Bidi supported	No

Enable high availability support (enableHASupport)

Do not change this property. It must be set to `true`.

Query timeout (QueryTimeOut)

This property specifies the maximum number of seconds a query can take for all SQL statements.

Table 76. Query timeout details

Required	No
Default	No default value
Unit of measure	Seconds
Property type	Integer

Table 76. Query timeout details (continued)

Usage	If the query takes longer than the number of seconds specified, the database generates an SQL exception that is captured. The associated message is logged in the log file. If a value is not specified, no timeout is set on the query.
Globalized	Yes
Bidi supported	No

Return business object even when the stored procedure result set is empty (ReturnDummyBOForSP)

This property specifies whether to return output parameters when the result set is empty.

Table 77. Return business object even when the stored procedure result set is empty details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	The Retrieve Stored Procedure (RetrieveSP) operation returns a result set. If the result set is empty and the ReturnDummyBOForSP property is set to `False`, no business objects are created, and the output parameters returned by the procedure call cannot be retrieved. However, if ReturnDummyBOForSP is `True`, a dummy business object is created and populated with values from output and input/output parameters in the corresponding attributes.
Globalized	Yes
Bidi supported	No

SQL query to verify the connection (PingQuery)

This property specifies the SQL query that is used to test valid connection to the database.

Table 78. Ping query details

Required	No
Property type	String
Default	No default value
Usage	This property contains the SQL query statement that you want to run to determine whether the adapter can connect to the database.
Globalized	No
Bidi supported	No

Activation specification properties

Activation specification properties are properties that hold the inbound event processing configuration information for an export.

You set activation specification properties using the external service wizard during adapter configuration and can change them using the WebSphere Integration

Developer assembly editor or, after deployment, the WebSphere Process Server or WebSphere Enterprise Service Bus administrative console.

The following table lists and describes the activation specification properties. A complete description of each property is provided in the sections that follow the table. For information about how to read the property detail tables in the sections that follow, see "Guide to information about properties" on page 178.

Table 79. Activation specification properties for Adapter for JDBC

Property name		
In the wizard	**In the administrative console**	**Description**
Additional JDBC driver connection properties	DriverConnectionProperties	Additional properties for connecting to the database using the JDBC driver, which are used in addition to the UserName and Password properties
Business object namespace	BusinessObjectNameSpace	The namespace for the business object definitions
Custom delete query	CustomDeleteQuery	The name of the query, stored procedure, or stored function that is run after each event is processed to delete records that can be deleted after the event is delivered
Custom event query	CustomEventQuery	The name of the query, stored procedure, or stored function that performs the polling for events
Custom update query	CustomUpdateQuery	The name of the query, stored procedure, or stored function that is run after each event is processed to prevent the event from being picked up for processing in a subsequent event cycle
Data source JNDI name	DataSourceJNDIName	The name of the JNDI data source to use to establish a connection to the database
Database URL	DatabaseURL	The database URL that is used to connect to the database
Database vendor	DatabaseVendor	The type of database that the adapter uses for special processing
Do not process events that have a timestamp in the future	FilterFutureEvents	Specifies whether the adapter filters out future events by comparing the timestamp on each event with the system time
Ensure once-only event delivery	AssuredOnceDelivery	Specifies whether the adapter provides assured once delivery of events
Event order by	EventOrderBy	The order in which events are retrieved and processed
Event query type	EventQueryType	Determines whether to use the standard event store or custom query
Event table name	EventTableName	Name of the database table that contains events generated by the database for inbound processing
Event types to process	EventTypeFilter	A delimited list of event types that indicates to the adapter which events it should deliver
Interval between polling periods	PollPeriod	The length of time that the adapter waits between polling periods
JDBC driver class	JDBCDriverClass	The class name of the JDBC driver that is used to connect to the database
Maximum connections	MaximumConnections	The maximum number of connections that the adapter can use for inbound event delivery

Table 79. Activation specification properties for Adapter for JDBC (continued)

Property name		
In the wizard	In the administrative console	Description
Minimum connections	MinimumConnections	The minimum number of connections that the adapter can use for inbound event delivery
Number of times to retry the system connection	RetryLimit	The number of times the adapter tries to reestablish an inbound connection after an error
Password	Password	Password for authorizing the user to retrieve events from the database
Poll quantity	PollQuantity	The number of events that the adapter delivers to the export during each poll period
Query time out	QueryTimeOut	The maximum number of seconds a query can take for all SQL statements
Retry interval if connection fails	RetryInterval	The length of time that the adapter waits between attempts to establish a new connection after an error during inbound operations
Return dummy business object for RetrieveSP	ReturnDummyBOForSP	Specifies whether to return output parameters when the result set is empty
SQL query to verify the connection	PingQuery	The SQL query used to test valid connection to the database
Stop the adapter when an error is encountered while polling	StopPollingOnError	Specifies whether the adapter stops polling for events when it encounters an error during polling
Stored procedure to run after polling	SPAfterPoll	The name of the stored procedure that you want to be run after each poll cycle
Stored procedure to run before polling	SPBeforePoll	The name of the stored procedure that you want to be run before the actual poll query is called
Type of delivery	DeliveryType	Determines the order in which events are delivered by the adapter to the export
User name	UserName	The database user name to use for inbound events

Additional JDBC driver connection properties (DriverConnectionProperties)

This property contains additional information for connecting to the database using the JDBC driver.

Table 80. Driver connection properties details

Required	No
Possible values	Database connection properties are database-specific.
Default	No default value
Property type	String
Usage	These connection properties are used in addition to the UserName and Password properties. They allow you to customize the database connection used by the adapter. Specify the connection properties as one or more *name*:*value* pairs, separated by the semicolon character (;).

Table 80. Driver connection properties details (continued)

Example	The following value of this property specifies a login time out interval, makes a read-only database connection, and sets the security mechanism: `loginTimeout:20;readOnly:true;securityMechanism:USER_ONLY_SECURITY`
Globalized	Yes
Bidi supported	No

Business object namespace (BusinessObjectNameSpace)

This property specifies the namespace for the business object definitions.

Table 81. Business object namespace property characteristics

Required	Yes
Default	`http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc`
Property type	String
Usage	This value is added as a prefix to the business object name to keep business object names logically separated.
Example	The following example shows the `Schema1Customer` business object with the default namespace: `http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/Schema1Customer`
Bidi supported	No

Custom delete query (CustomDeleteQuery)

Use this property to specify the SQL statement, stored procedure, or stored function to run after each event is processed to delete records that can be deleted after the event is delivered.

Table 82. Custom delete query details

Required	No
Default	No default value
Property type	String
Usage	Use this property to specify the SQL statement, stored procedure, or stored function to run when the EventQueryType property is set to `Dynamic`.
Globalized	Yes
Bidi supported	Yes

Custom event query (CustomEventQuery)

Use this property to specify the SQL statement, stored procedure, or stored function to run to poll for events in custom event processing.

Table 83. Custom event query details

Required	No
Default	No default value
Property type	String
Usage	Use this property to specify the SQL statement, stored procedure, or stored function to run during each poll cycle when the EventQueryType property is set to `Dynamic`.

Table 83. Custom event query details (continued)

Examples	In the following example, the custom event query runs an SQL statement that returns the event ID, object key, and object name of every record in the MY_EVENT_TABLE event store whose status column has the value `0`:
	`select event_id, object_key, object_name from MY_EVENT_TABLE where status = 0`
	The following example, for Oracle databases only, limits the returned event records to the value of the PollQuantity property:
	`select event_id, object_key, object_name from MY_EVENT_TABLEwhere status = 0 and rownum < POLL QUANTITY`
	The following example runs a stored procedure with two parameters:
	`CALL MY_EVENT_STORED_PROC (?,?)`
	The following example runs a stored function with one parameter and one return value:
	`? = CALL MY_EVENT_FUNCTION(?)`

Globalized	Yes
Bidi supported	Yes

Custom update query (CustomUpdateQuery)

Use this property to specify the SQL statement, stored procedure, or stored function to run after each event is processed so that the same event does not get picked up for processing in the subsequent event cycle.

Table 84. Custom update query details

Required	No
Default	No default value
Property type	String
Usage	Use this property to specify the SQL statement, stored procedure, or stored function to run when the EventQueryType property is set to `Dynamic`.
Globalized	Yes
Bidi supported	Yes

Data source JNDI name (DataSourceJNDIName)

This property specifies the name of the JNDI data source to use to establish a connection to the database.

Table 85. Data source JNDI name details

Required	No
Default	No default value
Property type	String

Table 85. Data source JNDI name details (continued)

Usage	2 Use this property to specify the JNDI name of a data source in WebSphere Process Server or 2 WebSphere Enterprise Service Bus that specifies connection information for the target database. To improve the performance of inbound or outbound operations, specify the name of a data source that is enabled for prepared statement caching. If the user name and password properties are also set, they override the user name and password in the data source. The properties for connecting to the database are used in the following order: 1. If the DataSourceJNDIName property is set, the adapter uses it to establish the connection to the database. If the UserName and Password properties are also set, they override the user name and password set on the data source. 2. If the DataSourceJNDIName property is not set, the adapter uses the XADataSourceName and XADatabaseName properties, if set, to establish the connection. 2 The DataSourceJNDIName property represents an XA or connection pool data source. If you 2 define a JNDI data source on the server that supports XA transactions and then specify that 2 data source when you configure the adapter, you can connect to any type of database that 2 supports XA transactions. If you use an XA data source and database, the adapter supports 2 XA transaction only for DB2 and Oracle databases. 3. If the DataSourceJNDIName, XADataSourceName, and XADatabaseName properties are not 2 set, then the adapter uses the DatabaseURL, JDBCDriverClass, UserName and Password properties to establish the connection. Do not confuse the data source JNDI name property with the JNDI name of a managed connection factory or activation specification on the server. The following list highlights important differences between the types of JNDI names: • Data source JNDI name – Specifies a connection to a database – Used instead of saving user name and password in adapter properties – Saved as an adapter property • JNDI name of the managed connection factory or activation specification – Specifies a connection to a managed connection factory or activation specification on the server – Used instead of specifying the value of each managed connection factory or activation specification property in the wizard – Saved as the connection target in the import file
Globalized	Yes
Bidi supported	No

Database URL (DatabaseURL)

This property specifies the JDBC driver-specific URL for creating a connection to the database.

Table 86. Database URL details

Required	Yes, unless one of the following sets of properties are set: • The DataSourceJNDIName property • The XADataSourceName and XADatabaseName properties
Default	No default value

Table 86. Database URL details (continued)

Property type	String
Usage	In the external service wizard, you compose the database URL by filling in database-specific fields. For example, the database URL for a DB2 database is composed of the database name, the server host name, and the database port number. In the administrative console, type the complete database URL value. If your database server supports IPv6, you can specify the host name portion of the database URL in IPv6 format. The properties for connecting to the database are used in the following order: 1. If the DataSourceJNDIName property is set, the adapter uses it to establish the connection to the database. If the UserName and Password properties are also set, they override the user name and password set on the data source. 2. If the DataSourceJNDIName property is not set, the adapter uses the XADataSourceName and XADatabaseName properties, if set, to establish the connection. 2 The DataSourceJNDIName property represents an XA or connection pool data source. If you 2 define a JNDI data source on the server that supports XA transactions and then specify that 2 data source when you configure the adapter, you can connect to any type of database that 2 supports XA transactions. If you use an XA data source and database, the adapter supports 2 XA transaction only for DB2 and Oracle databases. 3. If the DataSourceJNDIName, XADataSourceName, and XADatabaseName properties are not 2 set, then the adapter uses the DatabaseURL, JDBCDriverClass, UserName and Password properties to establish the connection. If you specify the host name as an IP address in IPv6 format, enclose the IP address in square brackets ([]).

Examples	The following are typical values for common database servers:
	
	DB2 universal (type 4) JDBC driver
	`jdbc:db2://www.example.com:50000/DB`
	
	DB2 universal JDBC driver with an IPv6 address
	`jdbc:db2://[fe80::20c:29ff:feea:1361%4]:50000/DB`
	
	DB2 Universal Database type 2 driver for local connection
	jdbc:db2:TEST
	
	DB2 Universal Database type 2 driver for remote connection
	jdbc:db2://www.example.com:50000/TEST
	
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`

Globalized	Yes
Bidi supported	Yes

Database vendor (DatabaseVendor)

This property specifies which type of database is used. The type is determined by the database vendor name.

Table 87. Database vendor details

Required	Yes

Possible values	`DB2`
	`MSSQLServer`
	`Oracle`
	`Others`

Table 87. Database vendor details (continued)

Default	No default value
Property type	String
Usage	Some SQL statements require special processing, which varied by database type. For example, the Struct and Array data types in Oracle require special processing. This property specifies which RDBMS is used, which determines the database type. Specify the value that corresponds to your database vendor, as follows: • `DB2`, for an IBM DB2 database • `Oracle`, for an Oracle database • `MSSQLServer`, for a Microsoft SQL Server database • `Others`, for all other database types For other database types, the adapter does not perform any special processing. Make sure that the correct driver is specified in the JDBCDriverClass property.
Globalized	No
Bidi supported	No

Delivery type (DeliveryType)

This property specifies the order in which events are delivered by the adapter to the export.

Table 88. Delivery type details

Required	No
Possible values	`ORDERED` `UNORDERED`
Default	`ORDERED`
Property type	String
Usage	The following values are supported: • `ORDERED`: The adapter delivers events to the export one at a time. • `UNORDERED`: The adapter delivers all events to the export at once.
Globalized	No
Bidi supported	No

Do not process events that have a timestamp in the future (FilterFutureEvents)

This property specifies whether the adapter filters out future events by comparing the timestamp on each event with the system time.

Table 89. Do not process events that have a timestamp in the future details

Required	Yes
Possible values	`True` `False`
Default	`False`
Property type	Boolean

Table 89. Do not process events that have a timestamp in the future details (continued)

Usage	If set to `True`, the adapter compares the time of each event to the system time. If the event time is later than the system time, the event is not be delivered. If set to `False`, the adapter delivers all events.
Globalized	No
Bidi supported	No

Ensure once-only event delivery (AssuredOnceDelivery)

This property specifies whether to provide ensure once-only event delivery for inbound events.

Table 90. Ensure once-only event delivery details

Required	Yes
Possible values	`True` `False`
Default	`True`
Property type	Boolean
Usage	When this property is set to `True`, the adapter provides assured once event delivery. This means that each event will be delivered once and only once. A value of `False` does not provide assured once event delivery, but provides better performance. When this property is set to `True`, the adapter attempts to store transaction (XID) information in the event store. If it is set to `False`, the adapter does not attempt to store the information. This property is used only if the export component is transactional. If it is not, no transaction can be used, regardless of the value of this property.
Globalized	No
Bidi supported	No

Event order by (EventOrderBy)

The order in which events are retrieved and processed.

Table 91. Event order by details

Required	No
Possible values	A comma-separated (,) list of column names in the event store, and the order attributes `asc` and `desc`
Default	`event_time, event_priority`
Property type	String
Usage	Specify a comma-separated list of column names from the event store, with the optional attributes for ascending or descending order.

Examples	To present events ordered first by time and then by priority, specify:
	`event_time, event_priority`
	
	To present events ordered first by object name in ascending order and then by event time in descending order, specify:
	`object_name asc, event_time desc`

Globalized	Yes

Table 91. Event order by details (continued)

Bidi supported	Yes

Event query type (EventQueryType)

This property specifies whether to use standard or custom query processing.

Table 92. Event query type details

Required	Yes
Possible values	`Standard` `Dynamic`
Default	`Standard`
Property type	String
Usage	The valid values are `Standard`, for standard event processing, and `Dynamic`, for custom event processing. If this property is set to `Dynamic`, the CustomEventQuery, CustomUpdateQuery, and CustomDeleteQuery properties are used. If this property is set to `Standard`, those properties are ignored.
Globalized	No
Bidi supported	No

Event table name (EventTableName)

This property specifies the name of the table in the target database that contains the event store, which is used for inbound processing.

Table 93. Event table name details

Required	Yes
Default	`WBIA_JDBC_EventStore`
Property type	String
Usage	Create the event store before starting to configure the adapter. For standard event processing, the event are generated by the database through a trigger or other mechanism. For custom query processing, the adapter saves events in the event store as it receives the result of the custom queries.
Globalized	Yes
Bidi supported	Yes

Event types to process (EventTypeFilter)

This property contains a delimited list of event types that indicates to the adapter which events it should deliver.

Table 94. Event types to process details

Required	No
Possible values	A comma-delimited (,) list of business object types
Default	`null`
Property type	String

Table 94. Event types to process details (continued)

Usage	Events are filtered by business object type. If the property is set, the adapter delivers only those events that are in the list. A value of `null` indicates that no filter will be applied and that all events will be delivered to the export. This property applies only to standard event processing. For custom event processing, the custom event query must perform any necessary filtering.
Example	To receive only events relating to the Customer and Order business objects, specify this value: `Customer,Order`
Globalized	No
Bidi supported	No

JDBC driver class (JDBCDriverClass)

This property specifies the class name of the JDBC driver that is used to connect to the database.

Table 95. JDBC driver class details

Required	Yes, if the DataSourceJNDIName property is not set
Possible values	Values are database-specific.
Default	No default value
Property type	String
Usage	In the external service wizard, the driver is specified for you if you select a common database software and driver combination, such as type 4 drivers for recent versions of IBM DB2, Oracle, and Microsoft SQL. For most type 2 drivers for most database software, you must provide the database class name. If you select a type 2 driver or a generic driver, you must type the JDBC driver class name. For example, for the DB2 Universal Database type 2 driver, the class name is `COM.ibm.db2.jdbc.app.DB2Driver` In the administrative console, type the database-specific name of the driver. If the DataSourceJNDIName property is set, this property is ignored.

Examples	**In the external service wizard:**
	• To connect to a DB2 database using the universal, or type 4, JDBC driver, select`IBM DB2 Universal`.
	• To connect to a DB2 database using the DB2 universal type 2 driver, select `Other`.
	• To connect to an Oracle 10 database using the type 4 driver, select `Oracle Thin Driver`.
	
	In the administrative console:
	
	DB2 Universal Database type 2 driver
	`COM.ibm.db2.jdbc.app.DB2Driver`
	
	DB2 Universal Database type 4 driver
	`com.ibm.db2.jcc.DB2Driver`
	
	Oracle Thin JDBC driver
	`oracle.jdbc.driver.OracleDriver`
	
	IBM Toolkit for Java remote driver for i5/OS
	`com.ibm.as400.access.AS400JDBCDriver`
	
	IBM WebSphere Connect JDBC driver for Microsoft SQL Server
	`com.ibm.websphere.jdbc.sqlserver.SQLServerDriver`

Table 95. JDBC driver class details (continued)

Globalized	No
Bidi supported	No

Maximum connections (MaximumConnections)

This property specifies the maximum number of connections that the adapter can use for inbound event delivery.

Table 96. Maximum connections details

Required	No
Default	1
Property type	Integer
Usage	Only positive values are valid. The adapter considers any positive entry less than 1 to be equal to 1. Typing a negative value or 1 for this property may result in run time errors.
Globalized	No
Bidi supported	No

Minimum connections (MinimumConnections)

This property specifies the minimum number of connections that the adapter can use for inbound event delivery.

Table 97. Minimum connections details

Required	No
Default	1
Property type	Integer
Usage	Only positive values are valid. Any value less than 1 is treated as 1 by the adapter. Typing a negative value or 1 for this property may result in run time errors.
Globalized	No
Bidi supported	No

Password (Password)

This property specifies the password for the database user name.

Table 98. Password details

Required	Yes, unless the DataSourceJNDIName or XADataSourceName properties are set
Default	No default value
Property type	String
Usage	2 If this property is set, it overrides the password specified on a data source on the server using either the DataSourceJNDIName or XADataSourceName properties.
Globalized	Yes
Bidi supported	Yes

SQL query to verify the connection (PingQuery)

This property specifies the SQL query that is used to test valid connection to the database.

Table 99. Ping query details

Required	No
Property type	String
Default	No default value
Usage	This property contains the SQL query statement that you want to run to determine whether the adapter can connect to the database.
Globalized	No
Bidi supported	No

Interval between polling periods (PollPeriod)

This property specifies the length of time that the adapter waits between polling periods.

Table 100. Interval between polling periods details

Required	Yes
Possible values	Integers greater than or equal to `0`.
Default	`2000`
Unit of measure	Milliseconds
Property type	Integer
Usage	The poll period is established at a fixed rate, which means that if running the poll cycle is delayed for any reason (for example, if a prior poll cycle takes longer than expected to complete) the next poll cycle will occur immediately to make up for the lost time caused by the delay.
Globalized	No
Bidi supported	No

Maximum events in polling period (PollQuantity)

This property specifies the number of events that the adapter delivers to the export during each poll period.

Table 101. Maximum events in polling period details

Required	Yes
Default	`10`
Property type	Integer
Usage	The value must be greater than 0. If this value is increased, more events are processed per polling period and the adapter may perform less efficiently. If this value is decreased, less events are processed per polling period and the adapter's performance may improve slightly.
Globalized	No
Bidi supported	No

Query timeout (QueryTimeOut)

This property specifies the maximum number of seconds a query can take for all SQL statements.

Table 102. Query timeout details

Required	No
Default	No default value
Unit of measure	Seconds
Property type	Integer
Usage	If the query takes longer than the number of seconds specified, the database generates an SQL exception that is captured. The associated message is logged in the log file. If a value is not specified, no timeout is set on the query.
Globalized	Yes
Bidi supported	No

Retry interval if connection fails (RetryInterval)

When the adapter encounters an error related to the inbound connection, this property specifies the length of time the adapter waits before trying to establish a new connection.

Table 103. Retry interval details

Required	Yes
Default	`2000`
Unit of measure	Milliseconds
Property type	Integer
Usage	Only positive values are valid. When the adapter encounters an error related to the inbound connection, this property specifies the length of time the adapter waits before trying to establish a new connection.
Globalized	Yes
Bidi supported	No

Number of times to retry the system connection (RetryLimit)

This property specifies the number of times the adapter tries to reestablish an inbound connection.

Table 104. Number of times to retry the system connection details

Required	No
Possible values	Positive integers
Default	`0`
Property type	Integer
Usage	Only positive values are valid. When the adapter encounters an error related to the inbound connection, this property specifies the number of times the adapter tries to restart the connection. A value of `0` indicates an infinite number of retries.

Table 104. Number of times to retry the system connection details (continued)

Globalized	Yes
Bidi supported	No

Return business object even when the stored procedure result set is empty (ReturnDummyBOForSP)

This property specifies whether to return output parameters when the result set is empty.

Table 105. Return business object even when the stored procedure result set is empty details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	The Retrieve Stored Procedure (RetrieveSP) operation returns a result set. If the result set is empty and the ReturnDummyBOForSP property is set to `False`, no business objects are created, and the output parameters returned by the procedure call cannot be retrieved. However, if ReturnDummyBOForSP is `True`, a dummy business object is created and populated with values from output and input/output parameters in the corresponding attributes.
Globalized	Yes
Bidi supported	No

Stop the adapter when an error is encountered while polling (StopPollingOnError)

This property specifies whether the adapter will stop polling for events when it encounters an error during polling.

Table 106. Stop the adapter when an error is encountered while polling details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	If this property is set to `True`, the adapter stops polling when it encounters an error. If this property is set to `False`, the adapter logs an exception when it encounters an error during polling and continues polling.
Globalized	No
Bidi supported	No

Stored procedure to run after polling (SPAfterPoll)

This property specifies the name of the stored procedure or stored function to run after each polling cycle.

Table 107. Stored procedure to run after poll details

Required	No
Default	No default value
Property type	String
Usage	The stored procedure takes one parameter for poll quantity.
Globalized	Yes
Bidi supported	Yes

Stored procedure to run before polling (SPBeforePoll)

This property specifies the name of any stored procedure or stored function to run before the actual poll query is called.

Table 108. Stored procedure to run before poll details

Required	No
Default	No default value
Property type	String
Usage	The stored procedure takes one parameter for poll quantity.
Globalized	Yes
Bidi supported	Yes

User name (UserName)

This property specifies the database user name for connecting to the database.

Table 109. User name details

Required	Yes, unless the DataSourceJNDIName or XADataSourceName properties are set
Default	No default value
Property type	String
Usage	2 If this property is set, it overrides the user name specified on a data source on the server using either the DataSourceJNDIName or XADataSourceName properties.
Globalized	Yes
Bidi supported	Yes

Globalization

WebSphere Adapter for JDBC is a globalized application that can be used in multiple linguistic and cultural environments. Based on character set support and the locale of the host server, the adapter delivers message text in the appropriate language. The adapter supports bidirectional script data transformation between integration components.

Globalization and bidirectional transformation

The adapter is globalized to support single- and multi-byte character sets and deliver message text in the specified language. The adapter also performs bidirectional transformation, which refers to the task of processing data that contains both left-to-right (Hebrew or Arabic, for example) and right-to-left (a URL or file path, for example) semantic content within the same file.

Globalization

Globalized software applications are designed and developed for use within multiple linguistic and cultural environments rather than a single environment. WebSphere Adapters, WebSphere Integration Developer, WebSphere Process Server, and WebSphere Enterprise Service Bus are written in Java. The Java runtime environment within the Java virtual machine (JVM) represents data in the Unicode character code set. Unicode contains encodings for characters in most known character code sets (both single- and multi-byte). Therefore, when data is transferred between these integration system components, there is no need for character conversion.

To log error and informational messages in the appropriate language and for the appropriate country or region, the adapter uses the locale of the system on which it is running.

Bidirectional transformation

Languages such as Arabic and Hebrew are written from right to left, yet they contain embedded segments of text that are written left to right, resulting in bidirectional script. There are multiple ways that a software application might display and process bidirectional script. WebSphere Process Server and WebSphere Enterprise Service Bus use the Windows standard format, but an enterprise information system exchanging data with WebSphere Process Server or WebSphere Enterprise Service Bus can use a different format. WebSphere Adapters transform bidirectional script data passed between the two systems so that it is accurately processed and displayed on both sides of a transaction.

Bidirectional format

WebSphere Process Server and WebSphere Enterprise Service Bus use the bidirectional format of ILYNN (implicit, left-to-right, on, off, nominal). This is the format used by Windows. If an enterprise information system uses a different format, the adapter converts the format prior to introducing the data to WebSphere Process Server or WebSphere Enterprise Service Bus.

Five attributes comprise bidirectional format. When you set bidirectional properties, you assign values for each of these attributes. The attributes and settings are listed in the following table.

Table 110. Bidirectional format 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

Table 110. Bidirectional format attributes (continued)

Letter position	Purpose	Values	Description	Default setting
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 N	Hindi Contextual Nominal	N

The adapter transforms data into a logical, left-to-right format before sending the data to WebSphere Process Server or WebSphere Enterprise Service Bus.

Using bidirectional properties

You can use multiple bidirectional properties to control the transformation of both content data and metadata. You can set special bidirectional properties to exclude either content data or metadata from bidirectional transformation, or to identify data that requires special treatment during a transformation.

The following table describes four types of bidirectional properties.

Table 111. Bidirectional property types

Property type	Data transformations
EIS	Controls the format for content data, or data that is sent by the enterprise information system, that is, the database.
Metadata	Controls the format for metadata, or data that provides information about the content data.
Skip	Identifies content or metadata to exclude from transformation.
Special Format	Identifies certain text, such as file paths or URLs, that require different treatment during the transformation process. Can be set for either content data or metadata.

You can set properties that control bidirectional transformation in the following areas:

- **Resource adapter properties:** These properties store default configuration settings, including the TurnBiDiOff property, which controls whether the adapter instance performs bidirectional transformation or not. Use the administrative console of the server to configure these properties.
- **Managed connection factory properties:** These properties are used at run time to create an outbound connection instance with an enterprise information system. After the managed connection factory properties are created, they are stored in the deployment descriptor.

- **Activation specification properties:** These properties hold the inbound event processing configuration information for a message endpoint. Set them when you use the external service wizard, or use the administrative console of the server.

Property scope and lookup mechanism

After you set values for bidirectional properties for an adapter, the adapter performs bidirectional transformations. It does so by using logic that relies on a hierarchical inheritance of property settings and a lookup mechanism.

Properties defined within the resource adapter are at the top of the hierarchy, while those defined within other areas or annotated within a business object are at lower levels of the hierarchy. So for example, if you set values for EIS-type bidirectional properties only for the resource adapter, those values are inherited and used by transformations that require a defined EIS-type bidirectional property, whether they arise from an inbound (activation specification) transaction or an outbound (managed connection factory) transaction.

However, if you set values for EIS-type bidirectional properties for both the resource adapter and the activation specification, a transformation arising from an inbound transaction uses the values set for the activation specification.

The processing logic uses a lookup mechanism to search for bidirectional property values to use during a transformation. The lookup mechanism begins its search at the level where the transformation arises and searches upward through the hierarchy for defined values of the appropriate property type. It uses the first valid value it finds. It searches the hierarchy from child to parent only; siblings are not considered in the search.

Properties enabled for bidirectional data transformation

WebSphere Adapter for JDBC has several configuration properties that are enabled for bidirectional data transformation.

The adapter enables the exchange of bidirectional data between a client application and the database, even if the data in the database is in a different bidirectional format than is used by the runtime environment. You can use bidirectional characters when configuring the adapter and in the application-specific information of your business objects. The following sets of properties and application-specific information are enabled for bidirectional support:

- Configuration properties
  - Activation specification properties
  - Connection properties for the external service wizard
  - Managed connection factory properties
- Application specific information
  - Business object level ASI
  - Operation level ASI
  - Attribute level ASI

The sections which follow list the specific configuration properties and application-specific information that are enabled for bidirectional transformation.

Connection properties used in the wizard

The following connection properties for theexternal service wizard are enabled for bidirectional script data transformation:

- User name
- Password

Managed connection factory properties

The following managed connection properties are enabled for bidirectional script data transformation:

- Additional JDBC driver connection properties
- Database URL
- Password
- User name
- XA database name

Activation specification properties

The following activation specification properties are enabled for bidirectional script data transformation:

- Custom delete query
- Custom event query
- Custom update query
- Additional JDBC driver connection properties
- Database URL
- Event order by
- Event table name
- Password
- Stored procedure to run before polling
- Stored procedure to run after polling
- User name

Business object application-specific information

The following business object application-specific information parameters are enabled for bidirectional script data transformation:

- TableName
- StatusColumnName
- SPName
- SelectStatement

Operation application-specific information

The following operation application-specific information parameters are enabled for bidirectional script data transformation:

- StoredProcedureName
- PropertyName in Parameters

Attribute application-specific information

The following attribute application-specific information parameters are enabled for bidirectional script data transformation:

- ColumnName

Adapter messages

View the messages issued by WebSphere Adapter for JDBC at the following location.

Link to messages: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/topic/com.ibm.wbit.610.help.messages.doc/messages.html

The displayed Web page shows a list of message prefixes. Click a message prefix to see all the messages with that prefix:

- Messages with the prefix CWYBC are issued by WebSphere Adapter for JDBC
- Messages with the prefix CWYBS are issued by the adapter foundation classes, which are used by all the adapters.

Related information

The following information centers, IBM Redbooks, and Web pages contain related information for the WebSphere Adapter for JDBC.

Samples and tutorials

The WebSphere Integration Developer online samples/tutorials gallery includes samples and tutorials to help you use WebSphere Adapters. You can access the online samples/tutorials gallery as follows:

- From the welcome page that opens when you start WebSphere Integration Developer. To see samples and tutorials for WebSphere Adapter for JDBC, click **Retrieve**. Then browse the displayed categories to make your selections.
- At this location on the Web: http://publib.boulder.ibm.com/bpcsamp/index.html.

Information resources

- The WebSphere Business Process Management information resources Web page includes links to articles, Redbooks, documentation, and educational offerings to help you learn about WebSphere Adapters: http://www14.software.ibm.com/webapp/wsbroker/redirect?version=pix&product=wps-dist&topic=bpmroadmaps
- The WebSphere Adapters library page includes links to all versions of the documentation: http://www.ibm.com/software/integration/wbiadapters/library/infocenter/

Information about related products

- WebSphere Business Process Management, version 6.1.0, information center, which includes WebSphere Process Server, WebSphere Enterprise Service Bus, and WebSphere Integration Developer information: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp
- WebSphere Adapters, version 6.0.2, information center: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.wsadapters602.doc/welcome_top_wsa602.html

- WebSphere Adapters, version 6.0, information center: http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/topic/com.ibm.wsadapters.doc/welcome_wsa.html
- WebSphere Business Integration Adapters information center: http://publib.boulder.ibm.com/infocenter/wbihelp/v6rxmx/index.jsp?topic=/com.ibm.wbi_adapters.doc/welcome_adapters.htm

developerWorks® resources

- WebSphere Adapter Toolkit
- WebSphere business integration zone

Support and assistance

- WebSphere Adapters technical support: http://www.ibm.com/software/integration/wbiadapters/support/
- WebSphere Adapters technotes: http://www.ibm.com/support/search.wss?tc=SSMKUK&rs=695&rank=8 &dc=DB520+D800+D900+DA900+DA800+DB560&dtm. In the **Product category** list, select the name of the adapter and click **Go**.

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.

© Copyright IBM Corp. 2006, 2007

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department 2Z4A/SOM1
294 Route 100
Somers, NY 10589-0100
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: (c) (your company name) (year). Portions of

this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning:

Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

IBM, the IBM logo, developerWorks, i5/OS, OS/400, Redbooks, Tivoli, ViaVoice, WebSphere, and z/OS are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

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

Microsoft and Windows are registered trademark of Microsoft Corporation in the United States, other countries, or both.

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

Other company, product, or service names may be trademarks or service marks of others.

This product includes software developed by the Eclipse Project (http://www.eclipse.org).

Index

© Copyright IBM Corp. 2006, 2007

Printed in USA