WebSphere® Adapters
Version 7 Release 0

WebSphere Adapter for JDBC User Guide
Version 7 Release 0

IBM®

WebSphere® Adapters
Version 7 Release 0

WebSphere Adapter for JDBC User Guide
Version 7 Release 0

IBM®

Note

Before using this information and the product it supports, read the information in “Notices” on page 285.

Contents

Chapter 1. Overview of WebSphere Adapter for JDBC 1
What is new in this release 1
Hardware and software requirements 2
Technical overview of WebSphere Adapter for JDBC 2
Outbound processing 4
Inbound processing. 17
Business objects 22
Stored procedure overview 38
Stored procedure business object overview . . . 38
Stored procedures used in place of or in addition to operations 40
Stored functions overview 45
Query business object overview 46
Batch SQL business object overview 46
The external service wizard 47
Support for the Log and Trace Analyzer 47
Business faults 48

Chapter 2. Planning for adapter implementation 49
Before you begin 49
Security 49
Support for protecting sensitive user data in log and trace files 49
User authentication. 50
Deployment options 51
WebSphere Adapters in clustered environments . . 53
Support for prepared statement caching 55
Migrating to version 7.0 of WebSphere Adapter for JDBC 55
Migration considerations 55
Performing the migration. 57
Upgrading but not migrating a project 59
Migrating WebSphere Business Integration applications for use with Version 7.0 WebSphere Adapters 59
Roadmap for migrating applications from WebSphere InterChange Server 60
Migration considerations for WebSphere Business Integration adapters 61
Migrating application artifacts from WebSphere InterChange Server 62
Migrating adapter-specific artifacts 63
Changes to the import, export, and WSDL files after migration 66

Chapter 3. Samples and tutorials . . . 67

Chapter 4. Configuring the module for deployment. 69
Road map for configuring the module 69
Creating the event store 70
Creating an authentication alias 71
Creating the project. 72
Adding external software dependencies 73
Setting connection properties for the external service wizard 75
Configuring the module for outbound processing . 78
Discovering database objects. 78
Selecting and configuring business objects . . . 82
Setting global properties for operations and creating wrapper business objects 100
Setting deployment properties and generating the service 103
Completing the configuration 110
Configuring the module for inbound processing 112
Discovering database objects 112
Selecting and configuring business objects. . . 115
Setting global properties for operations and creating wrapper business objects 124
Setting deployment properties and generating the service 126
Completing the configuration 134

Chapter 5. Modifying artifacts 137
Modifying service import 137
Modifying service export 139

Chapter 6. Changing interaction specification properties using the assembly editor 141

Chapter 7. Deploying the module . . . 143
Deployment environments 143
Deploying the module for testing. 143
Adding external dependencies. 143
Generating and wiring a target component for testing inbound processing 145
Preparing to test outbound operations 146
Adding the module to the server 147
Testing the module for outbound processing using the test client 147
Deploying the module for production 148
Adding external software dependencies on the server 148
Installing the RAR file (for modules using stand-alone adapters only) 149
Exporting the module as an EAR file 152
Installing the EAR file 153

Chapter 8. Administering the adapter module 155
Changing configuration properties for embedded adapters 155
Setting resource adapter properties for embedded adapters 155

© Copyright IBM Corp. 2006, 2009

Setting managed (J2C) connection factory properties for embedded adapters 157
Setting activation specification properties for embedded adapters 159
Changing configuration properties for stand-alone adapters 161
Setting resource adapter properties for stand-alone adapters 161
Setting managed (J2C) connection factory properties for stand-alone adapters 162
Setting activation specification properties for stand-alone adapters 163
Starting the application that uses the adapter . . . 164
Stopping the application that uses the adapter . . 165
Monitoring performance using Performance Monitoring Infrastructure 166
Configuring Performance Monitoring Infrastructure 166
Viewing performance statistics 168
Enabling tracing with the Common Event Infrastructure (CEI) 169

Chapter 9. Troubleshooting and support 171
Configuring logging and tracing 171
Configuring logging properties 171
Changing the log and trace file names 173
First-failure data capture (FFDC) support 173
Solutions to common problems 174
Self-help resources. 195

Chapter 10. Reference 197
Business object information. 197
Business object attributes 197
Attribute application-specific information . . . 199
Business object-level application-specific information 206
Naming conventions 210
Configuration properties 212
Outbound configuration properties 212
Inbound configuration properties. 240
Globalization 277
Globalization and bidirectional transformation 277
Properties enabled for bidirectional data transformation 280
Fault business objects. 282
Adapter messages 283
Related information 283

Notices 285
Programming interface information 287
Trademarks and service marks 287

Index 289

Chapter 1. Overview of WebSphere Adapter for JDBC

With WebSphere Adapter for JDBC, you can create integrated applications that can interact and 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. The JRE version required by the JDBC driver must be lower than or equal to the JRE version in the runtime environment. Examples of such servers include IBM® DB2®, Oracle, Microsoft® SQL 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.

What is new in this release

This version includes several new features that enhance the business flexibility, user experience, and performance of the adapter.

This information is also available at the WebSphere Adapters product support Web site http://www.ibm.com/software/integration/wbiadapters/support/, which is periodically updated with the latest information.

WebSphere Adapter for JDBC, version 7.0, supports the following new features:

- Modifying artifacts:
  - Facilitates incremental updates to existing modules.
  - Eliminates the need to create a module at every incremental development.
- User-defined types (STRUCTS) for Oracle table and query business objects.
- Generic business objects to support dynamic result sets from stored procedures.
- Support for multiple tables as source for inbound events.

© Copyright IBM Corp. 2006, 2009

- RetrieveAll operation to support:
  - Specific value to return all records from the database.
  - Empty result sets as a non-exception condition.
- Mapping SQL data types `Date`, `Time`, and `Timestamp` to `string` or `date`, `time`, `dateTime` XSD types as required in the business objects.
- Two foreign keys can now be defined in one field to enhance support for hierarchical business object relationship.
- Support for repeated attempts to establish connection with the database if the connection fails during outbound processing.

The following property is deprecated from this release. However, this property will still be supported for the earlier releases until it is made obsolete in the future.

- DataSourceJNDIName property is deprecated in outbound processing. For more information, see "Deprecated features" on page 57.

Hardware and software requirements

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

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

Additional information

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

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

Technical overview of WebSphere Adapter for 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 stored 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 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.

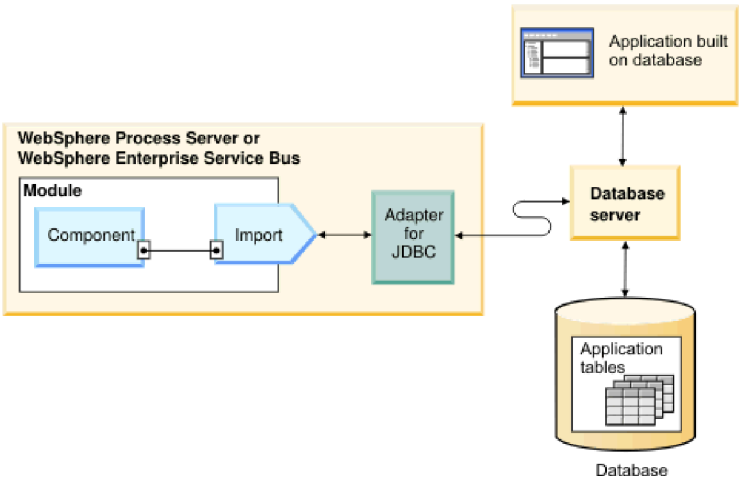


Figure 1. Processing outbound requests

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 4 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.

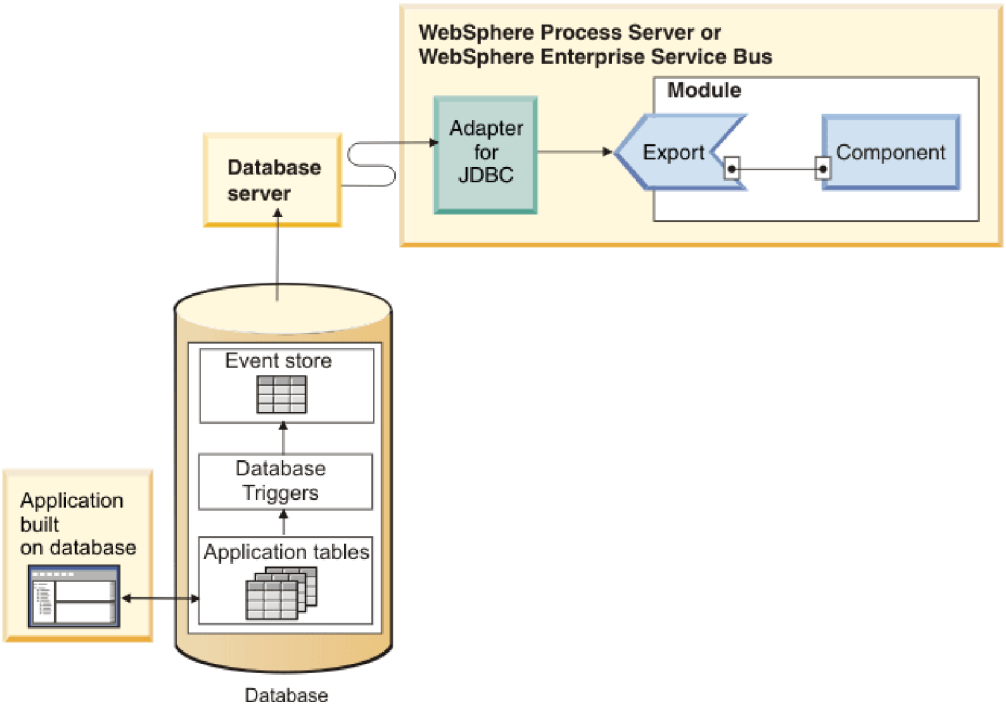


Figure 2. Processing inbound events

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 a 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 and connector ID. 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 query the existence of a record in the database or 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 wanted 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

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
	Exists	Not applicable	Not applicable
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. The transaction is managed and performed by the database.
- An *XA transaction* is one in which the transaction can span multiple heterogeneous databases. It uses a global, or two-phase commit, protocol. The transaction manager coordinates the transaction.

XA transactions

Choose one of these methods to configure the adapter for XA transactions:

- Set the connection type as XADataSourceJNDI, and specify the XA data source JNDI using the "XA DataSource JNDI name (XADataSourceJNDIName)" on page 236 property.
- Set the connection type as XAConnectionProps, and specify XA database connection properties.

  Note: Using XA database connection properties, you can configure the adapter to support global transactions for Oracle and DB2 databases only.

For information about how to configure XA transaction, see "Setting deployment properties and generating the service" on page 103.

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 application-specific information for the attribute. If an attribute has an associated database sequence or counter, the value generated by the adapter overwrites any value passed 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. Because the Retrieve operation uses only the primary key, the primary key must be provided for the top-level business object. Other columns are ignored.

   Note: Only the primary key provided in top-level business is used to construct the WHERE clause in the SQL statement.
   - 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.
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. The ownership attribute is ignored. The adapter uses the foreign key in the top-level business object to retrieve the child business object. All values configured in the child business object are ignored. If a foreign key is configured in the top-level business object and no records are found in the child business object, the operation returns a null business object.

   Note: The single-cardinality child business objects are processed based on their order of occurrence in the business object and after the parent business object is processed.

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 adapter uses different processes depending on whether the RetrieveAll operation is for database table business objects or for user-specified SQL business objects.

For database table business objects

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. All values specified in the top-level business object are used. The settings in the child business object are ignored. 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 business object for all the retrieved rows. The name of the container business object is the name of the business object 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.

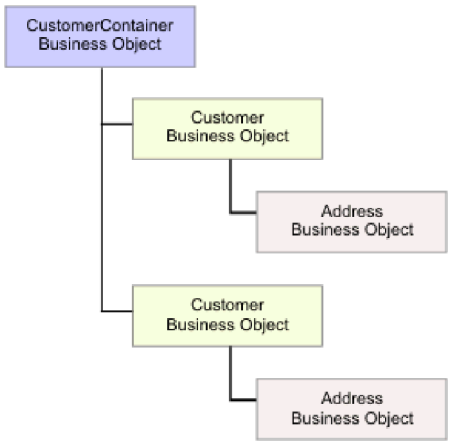


Figure 3. Structure of the business object returned in a RetrieveAll operation without optional business graphs

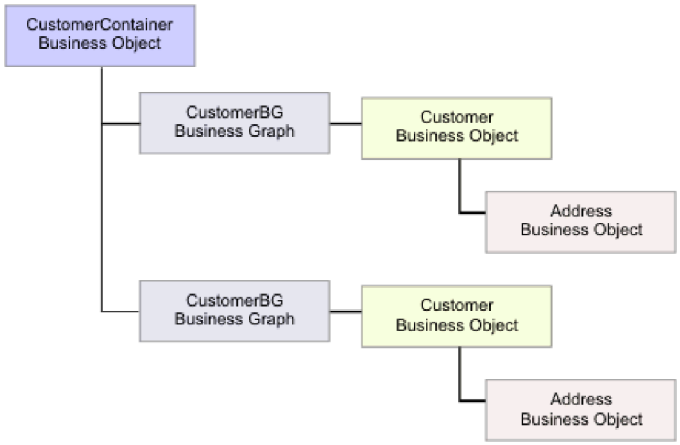


Figure 4. Structure of the business object returned in a RetrieveAll operation with optional business graphs

The following errors can result from a RetrieveAll operation:

- `RecordNotFoundException`– This exception is generated when one or more populated business objects in the input object does not exist in the enterprise information system and if the managed connection factory property for the ErrorOnEmptyResultset property is set to True. To get an empty result set without the RecordNotFoundException, you can either modify the .import file and set the "Throw exception when no records are found (ErrorOnEmptyResultSet)" on page 230 property to False, or after you deploy the outbound application on WebSphere Process Server, configure the MCF property "Throw exception when no records are found (ErrorOnEmptyResultSet)" on page 230 to False.
- `MatchesExceededLimitException` – This exception is generated when the number of matching records in the database exceeds the value of the "Maximum records for RetrieveAll operation" on page 239 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.

Note: If the "Maximum records for RetrieveAll operation" on page 239 property is set to a large number, problems can occur due to a lack of sufficient memory, depending upon the size and number of business objects returned.

- `EISSystemException` – This exception is generated when one or more unrecoverable errors are reported by the database (the enterprise information system)

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 will look like that shown in Figure 5.

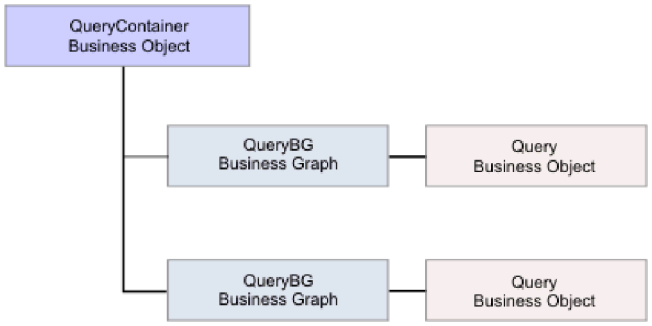


Figure 5. User-specified query business objects

If you are not using optional business graphs, the hierarchy will look like that shown in Figure 6.

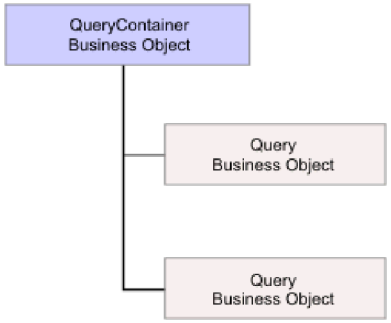


Figure 6. User-specified query business objects

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. Determines 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 10.
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 object 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 the ccode column 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 an 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. Typically, 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

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, as well as wrapper 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 with an Execute operation.

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, following is 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, retArr)`.
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.

Exists operation:

The Exists operation determines whether the database contains records that match the attributes set in a business object.

You can use both key and non-key attributes in the selection criteria.

Note: When using the external service wizard to discover table objects in a database, you can select multiple tables and add those tables to the selected objects area of the Object Discovery and Selection screen. However, you cannot use the

external service wizard to link or join the tables that you selected. If the goal of your business application requires the table business object to perform an Exists operation on joined tables, you need to join the tables in the database to create a view of the joined tables. After you have created a view of the joined tables, you can then perform discovery on the view. The Exists operation would be supported on this view.

To process the Exist operation and send the results based on the specified business object attributes, the adapter performs the following actions:

1. The adapter receives a table business object from the import. This business object can be flat (simple with no child business objects) or hierarchical (complex, containing one, or more child business objects).

   If the business object is hierarchical, it is only for the top-level business object (the individual business object at the top of a hierarchical business object) for which the adapter builds the query.

   Note: The input business object that supports the Exists operation varies depending on the business object type. In addition to being supported by the table business object, the Exists operation is also supported by the views business object and the synonyms and nicknames business object.
2. The adapter uses the table business object to generate an SQL SELECT statement that it sends to the server.

   The SQL SELECT statement used is as follows:

   ```
   select count(*) from TABLENAME where column1=? AND column2=?
   ```

   Here is a sample SQL statement for our example:

   ```
   select count(*) from CUSTOMER where fname='John' AND lname='Smith'
   ```

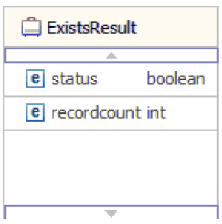
   In this case, the SQL statement specifies *non-primary key* attributes `fname` and `lname`, with the assigned values of `John` and `Smith`.

   The adapter includes the attribute information from the table business object in the *where* clause of the SQL query.
3. The database server runs the SQL query and sends the results back to the adapter.
4. The adapter obtains the results of the SQL query from the database server and sets the `recordcount` and `status` attributes on the **ExistsResults** business object.

   For example, if the Exists operation determined there are two records that match the attribute and value settings in the business object, the adapter sets `status=true` and `recordcount=2`.

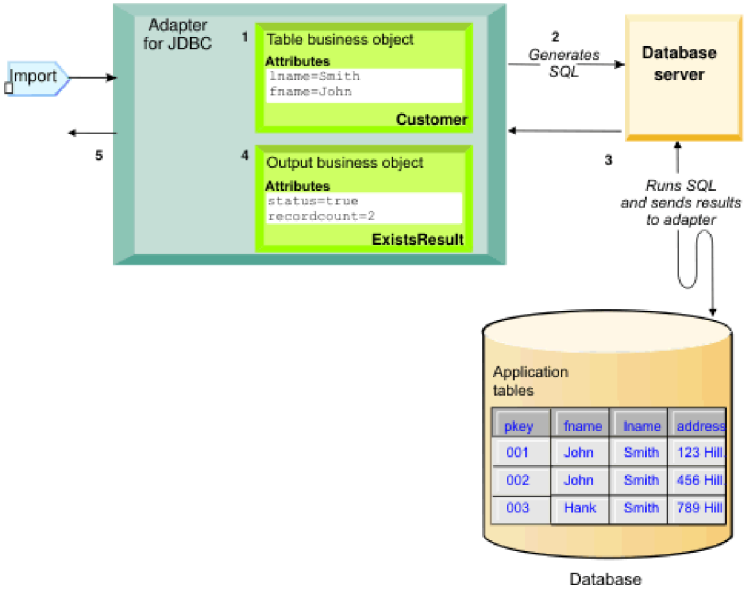
   If a record with the specified attributes is not found, the `status` output parameter is **false** and the `recordcount` output parameter is **0**.

   The following screen capture illustrates the structure of the ExistsResult business object returned from an Exists operation.

5. The adapter returns the ExistsResult business object to the caller.

The following illustration shows how the adapter processes a table business object with an Exists operation.



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 6.1.x 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*/scripts directory, where *version* identifies the version of the adapter, for example, 7.0.0.0. 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 then 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 timestamp on each event. If an event is set to occur in the future, it will not be processed until that time.

Custom event processing

In custom event processing, you provide the SQL queries or stored procedures that 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.

Note: When you are using custom event processing, set the property "Ensure once-only event delivery (AssuredOnceDelivery)" on page 263 to `True`.

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. 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. 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 the syntax 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 track the 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 column 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 column. 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 Retrieve 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 and 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 in 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 ID 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`.
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 the event was generated. The format is `mm/dd/yyyy hh:mm:ss`.

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

Column name	Type	Description
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.
connector_ID	String	The unique identifier for the adapter instance that receives a specific 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 adapter 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

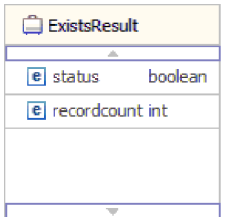
Some business objects do not represent database objects. These business objects include:

- 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, as well as allow you to group multiple stored procedures into a single business object.

Adapters use some business objects for output. These business objects include:

- Container business object, which contains the output from a RetrieveAll operation.

- ExistsResult business object, which contains the output from an Exists operation.



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 must 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 in the primary key attribute when it retrieves table business objects. The adapter 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 discover and configure that business object. Specify a column that contains unique data, such as a sequence or identity column. Identity columns (known as *serial columns* in Informix;JDBC adapter supports both serial and serial8) provide a way for the database to automatically generate a unique numeric value for each row in a table. A table can have a single column that is defined with the identity attribute. Examples of an identity column include order number, employee number, stock number, and incident number. Identity columns can be defined for tables in DB2, Informix and Microsoft SQL Server only.

Note: When you run the discovery process against a table in either a DB2 or Microsoft SQL Server database, and that table defines a column as an identity column, the generated business object for that table does not include the Unique

Identifier attribute of the identity column. In this case, you need to edit the generated business object by adding the attribute to the application-specific information manually. You can do this through the assembly editor in WebSphere Integration Developer. You do not need to add the attribute for the Unique Identifier manually if you ran the discovery process against a table in an Informix database. For Informix, the generated business object includes the Unique Identifier attribute of the serial column.

Table and view business objects support the Create, Update, Delete, Retrieve, RetrieveAll, Exists, and ApplyChanges outbound operations. When running an Exists operation on a hierarchical table business object, only the top-level business object is queried.

Figure 7 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.

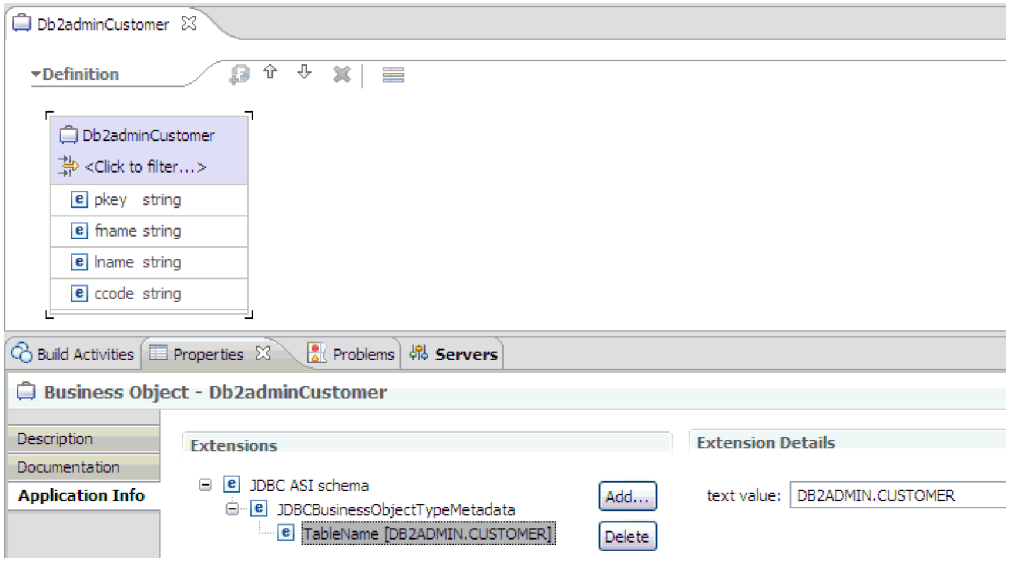


Figure 7. A table business object with no child.

Figure 8 on page 25 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.

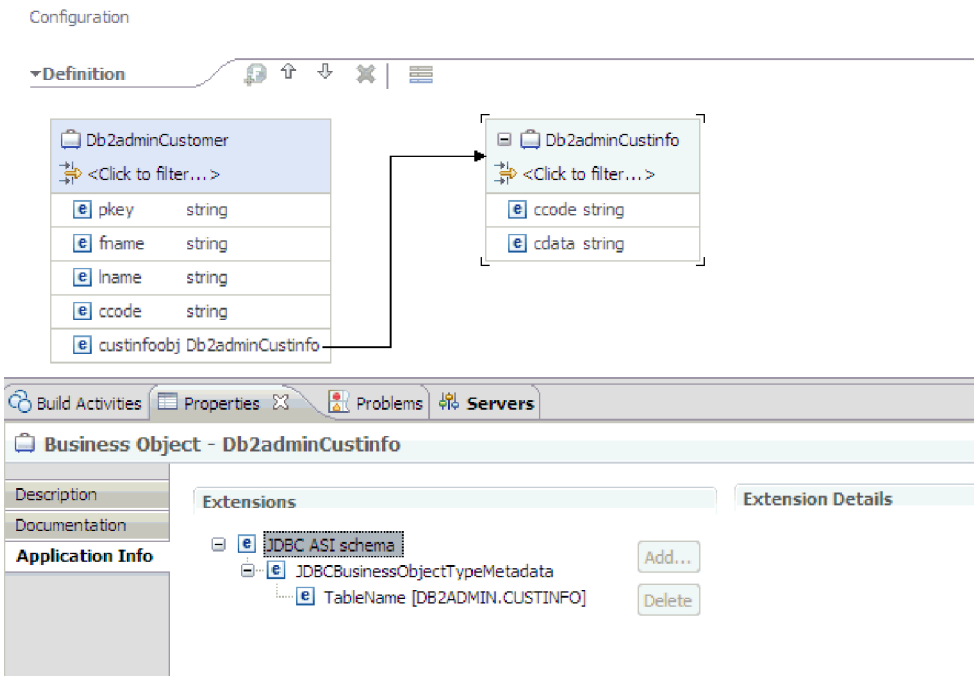


Figure 8. A table business object with one child business object.

For Oracle databases, the adapter supports complex data types such as array, table, structure, or nested structure in table business objects. The type name and the sub attribute details are automatically discovered and displayed for these types. The adapter processes these data types as child business objects of the table business object.

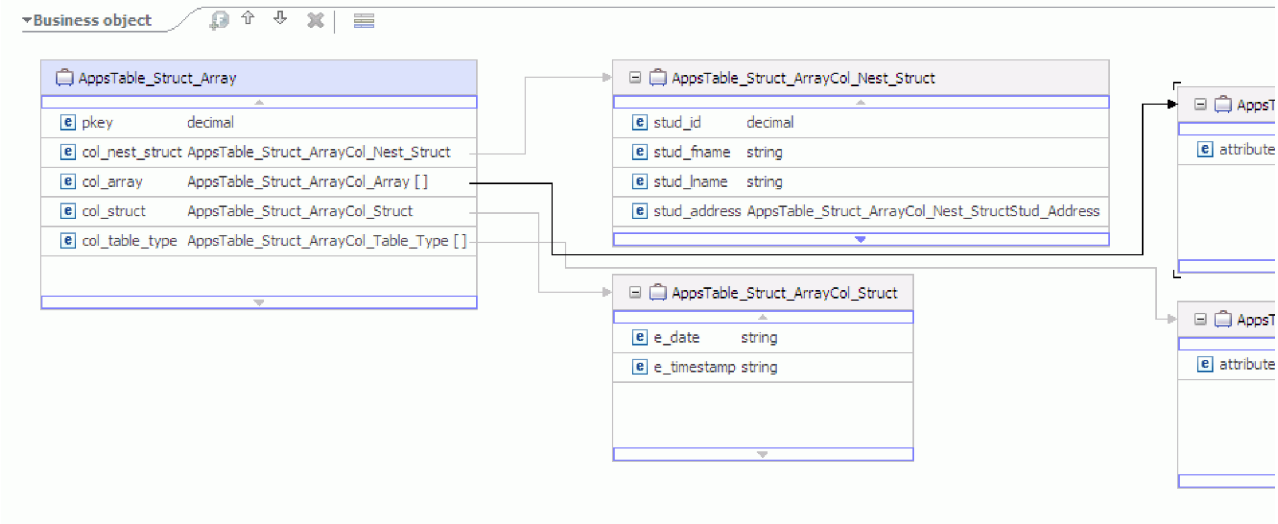


Figure 9. An Oracle table business object having complex data types as columns

For stored procedure and stored function business objects

In a business object for a stored procedure or stored function, all 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 show 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.

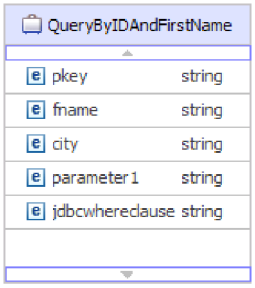


Figure 10. The attributes of a query business object

This figure shows the application-specific information for the query business object example. The SelectStatement application-specific information contains the SELECT statement.
For Oracle databases, the adapter supports complex data types such as array, table,

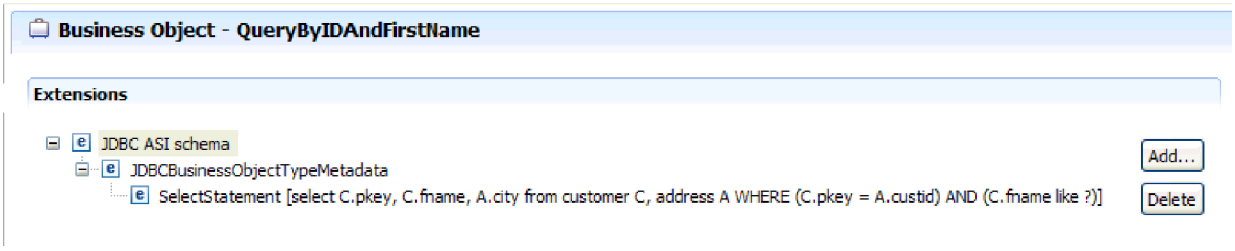


Figure 11. The SELECT statement is saved in the business object application-specific information

structure, or nested structure in the query result of the business object. The adapter does not support these complex types as parameters in batch and query business objects.

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 12 on page 30 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.

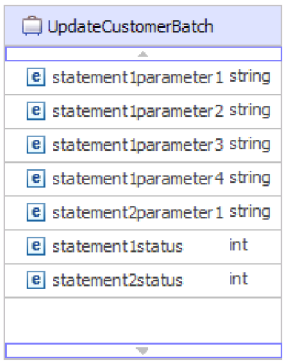


Figure 12. A batch SQL business object with two SQL statements

The business object has an attribute for each parameter in each statement, including statement1parameter1, statement2parameter1, and so on. It also has an attribute for 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 13.

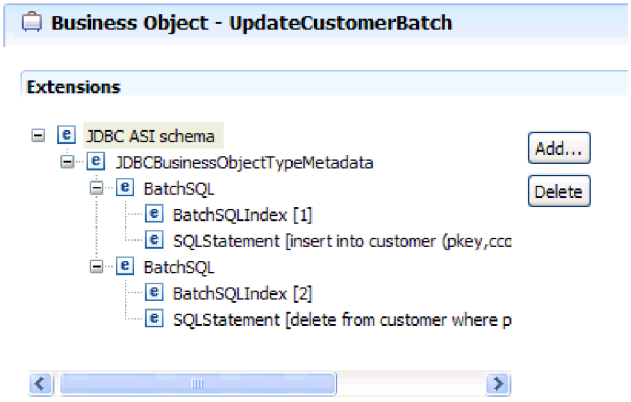


Figure 13. The application-specific information of a batch SQL business object

For wrapper business objects

A wrapper business object enables 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.

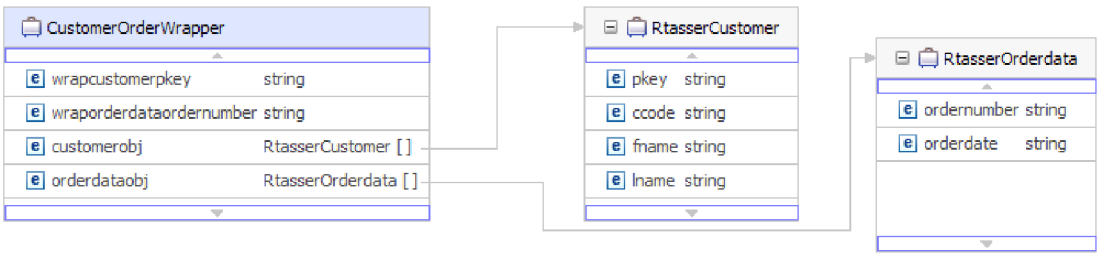


Figure 14. A wrapper business object that contains two table business objects

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 7.0, 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 7.0

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.

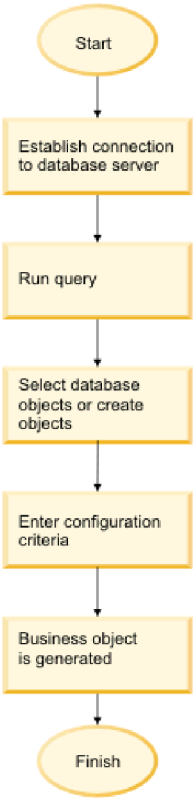


Figure 15. How business objects are created

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/v7r0mx/index.jsp

Business object hierarchies

Define the relationships between database tables using parent-child relationships and data ownership in hierarchical business objects. Unrelated tables can be 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. 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. 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 16 on page 34 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.

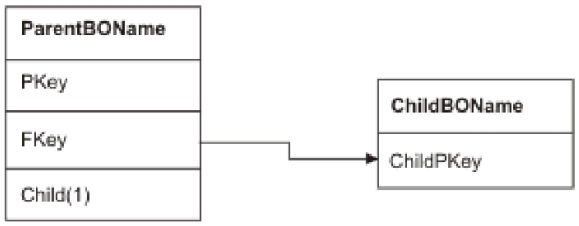
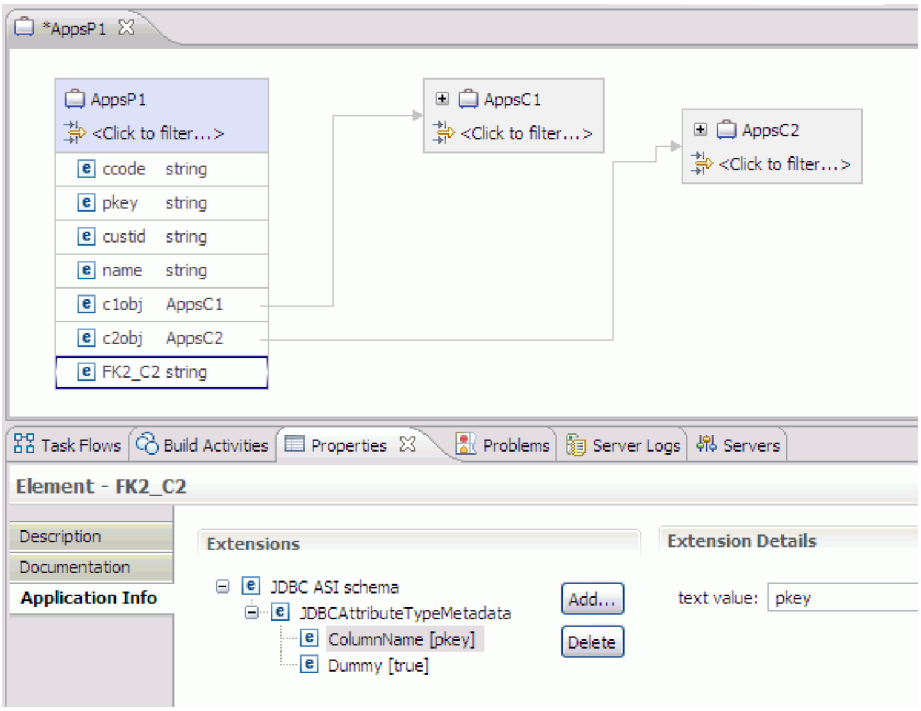


Figure 16. Typical single-cardinality relationship

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. If the parent has more than one child and the foreign key of the children reference the same column in the parent, edit the parent business object, add as many dummy columns as the children referencing the parent. For each column, add the CoulmnName, ForeignKey, and Dummy application-specific information, as shown in the example below.

1. Set the ColumnName ASI to the column referenced by the child in the parent.



2. Set the corresponding ForeignKey ASI
3. Set the Dummy ASI to "true"
4. Open the *.xsd file, and add the attribute `nillable="true"`

```
<xsd:element minOccurs="0" name="fk_custinfo_edu" nillable="true" type="xsd:string">
  <xsd:annotation>
    <xsd:appinfo source="http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/metadata">
      <metadata:JDBCAttributeTypeMetadataxmlns:metadata="http://www.ibm.com/xmlns/prod/wek
      <metadata:ColumnName>custid</metadata:ColumnName>
      <metadata:ForeignKey>ps_sfk_custinfo_eduobj/pkey</metadata:ForeignKey>
      <metadata:Dummy>true</metadata:Dummy>
      </metadata:JDBCAttributeTypeMetadata>
    </xsd:appinfo>
   </xsd:annotation>
</xsd:element>
```

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 differs 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 17 on page 37 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 Child(1) in the ParentBOName box is that attribute that represents the array of child business objects.

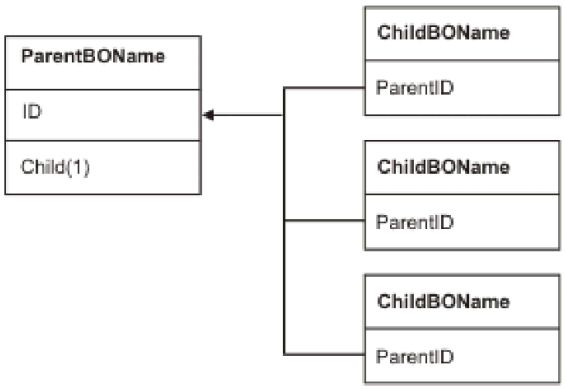


Figure 17. Multiple cardinality business object relationship with N>1

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 18 shows this type of relationship.

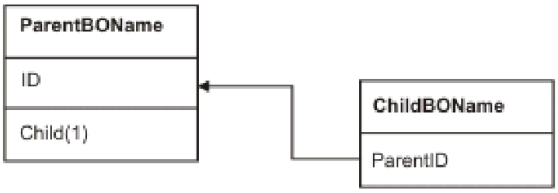


Figure 18. Multiple cardinality relationship with N=1

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.

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
- By creating a wrapper business object for stored procedures

  A wrapper business object for stored procedure allows you to group multiple stored procedures or execute the same stored procedure multiple times. The business objects corresponding to the stored procedures are added as multi-cardinality children of the wrapper stored procedure.

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 the 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.

Note: If the database schema has multiple stored procedures with the same name, and the stored procedures have different parameters, the external service wizard cannot identify which stored procedure is selected. Hence, the stored procedures must have unique names.

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 or output parameters, or return value parameters that are complex data types such as Struct, Array, or result set, you need to select the corresponding 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 a 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 result set, 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 are handled by the adapter run time.

During discovery and at run time, the WebSphere Adapter for JDBC expects the returned result set from the stored procedure execution to contain columns with names. Some stored procedures return result set with unnamed columns. For example, a stored procedure with the SQL statements like the examples that follow return result set with unnamed columns:

```
SELECT COUNT(*) FROM EMPLOYEE;
SELECT 111,222,333 FROM CUSTOMER;
```

Oracle processes such SQL SELECT statements by assigning "dummy" names to the table columns in the returned result set- like `count(*)` or `d1, d2, d3` for the respective SELECT statement in the examples shown.

If the returned result set contains table columns with no names (because the database did not assign dummy names), the adapter creates dummy names for such columns.

Dummy column names, generated by either the database or by the adapter, are assigned to the attributes of the stored procedure business object.

The behavior (by the adapter or by the database) of assigning dummy names to unnamed table columns ensures that the stored procedure runs successfully during discovery and at run time.

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 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`

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

Property name	Type	Description
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 run time. 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 on page 41 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 44.

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 it 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 the stored procedure type 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
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 the corresponding operation application-specific information property groups are also removed.

Table 6. Stored procedure type element characteristics (continued)

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

This value determines whether the stored procedure returns a result set or not. 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.

Table 8. Result set element characteristics

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

Table 9. Parameters element characteristics (continued)

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	For 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 is 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 run time 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 must be returned as an output parameter, not as a returned value. The type of the output parameter is set as RS, to indicate that this parameter is used to return a result set. **Note:** 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

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

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

```
        <jdbcasi:StoredProcedureName>RT.CUSTINFO</jdbcasi:StoredProcedureName>
          <jdbcasi:Parameters>
            <jdbcasi:Type>IP</jdbcasi:Type>
            <jdbcasi:PropertyName>pkey</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: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 like stored procedures except that they always return a value. The adapter 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 refers to a Java stored function or procedure, which the adapter does not support.

For DB2 databases, the adapter supports stored procedures that returns 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
```

A stored procedure call 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 about ReturnValue, see "Business object-level application-specific information" on page 206.

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 the 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 attributes are used to generate one dynamic WHERE clause at run time to replace the default WHERE clause.

For example, assume that 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 StatementStatus. 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 generates the exception. If one of the statements in a batch UPDATE fails to run properly, the JDBC driver generates the java.sql.BatchUpdateException. When this exception is generated, the adapter rolls back the transaction so that none of the SQL statements are committed to 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.

Support for the Log and Trace Analyzer

The adapter creates log and trace files that can be viewed with the Log and Trace Analyzer.

The Log and Trace Analyzer can filter log and trace files to isolate the messages and trace information for the adapter. It can also highlight the adapter's messages and trace information in the log viewer.

The adapter's component ID for filtering and highlighting is a string composed of the characters `JDBCRA` plus the value of the adapter ID property. For example, if the adapter ID property is set to `001`, the component ID is `JDBCRA001`.

If you run multiple instances of the same adapter, ensure that the first seven characters of the adapter ID property are unique for each instance so that you can correlate the log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate the log and trace information to a particular instance of an adapter. For example, when you set the adapter ID property of two instances of WebSphere Adapter for JDBC to `001` and `002`. The component IDs for those instances, `JDBCRA001` and `JDBCRA002`, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. If you set the adapter ID properties of two instances to `Instance01` and `Instance02`, you will not be able to examine the log and trace information for each adapter instance because the component ID for both instances is truncated to `JDBCRAInstanc`.

For outbound processing, the adapter ID property is located in both the resource adapter and managed connection factory property groups. If you update the adapter ID property after using the external service wizard to configure the adapter for outbound processing, be sure to set the resource adapter and managed connection factory properties consistently. It prevents inconsistent marking of the log and trace entries. For inbound processing, the adapter ID property is located only in the resource adapter properties, so this consideration does not apply.

For more information about the adapter ID property, see “Adapter ID (AdapterID)” on page 221. For more information about the Log and Trace Analyzer, see http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/org.eclipse.hyades.log.ui.doc.user/concepts/cltaviews.htm.

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, and are caused by a business rule violation or a constraint violation.

Although WebSphere Process Server or 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 primaryKeySet attributes as shown in Figure 19.

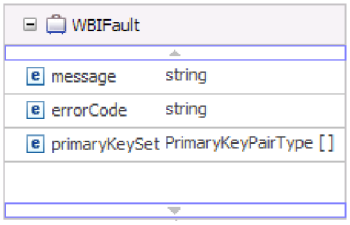


Figure 19. The structure of the WBIFault business object

Chapter 2. Planning for adapter implementation

Before using WebSphere Adapter for JDBC, make sure that 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 must possess a thorough understanding of the 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 must 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 such as the WebSphere Process Server or WebSphere Enterprise Service Bus. You must 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 must 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.

Support for protecting sensitive user data in log and trace files

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

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

To protect the data in such situations, the adapter lets you specify whether you want to prevent confidential user data from displaying in the adapter log and trace

© Copyright IBM Corp. 2006, 2009

files. You can select this option in the external service wizard or change the HideConfidentialTrace property. When this property is enabled, the adapter replaces the sensitive data with XXX's.

See “Managed connection factory properties” on page 225 for information about this optional property.

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

- The contents of a business object
- The contents of the object key of the event record
- User name, Password, Environment, and Role
- The URL used to connect to the database

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

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

User authentication

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

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

- When the 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 the discovery process, and then reuses it as the default values of the adapter properties that specify the connection information used at run time. You can use a different user name and password while running the wizard than you use when the application is deployed to the server. You can even connect to a different 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 data entry, or authentication alias, created with the Java Authentication and Authorization Service (JAAS) feature of Java 2 security is a robust, secure way to deploy applications. An administrator creates the authentication alias that is used by one or more applications that need to access a system. The user name and password can be known only to that administrator, who can change the password in a single place when a change is required.

Deployment options

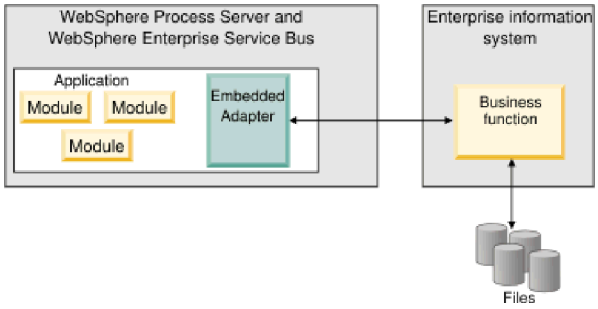
There are two ways to deploy the adapter. You can either embed it as part of the deployed application, or you can deploy it as a stand-alone RAR file. The requirements of your environment affect the type of deployment option you choose.

The deployment options are described below:

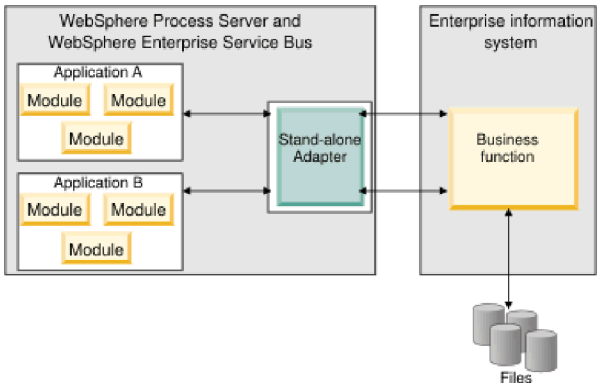
- **With module for use by single application**: With the adapter files embedded in the module, you can deploy the module to any application server. Use an embedded adapter when you have a single module using the adapter or if multiple modules need to run different versions of the adapter. Using an embedded 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.



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 affects how the adapter is used in the run time environment, as well as how the properties for the adapter are displayed on the administrative console.

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

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

Considerations for embedding an adapter in the application

Consider the following items if you plan to embed the adapter with your application:

- An embedded adapter has class loader isolation.

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

Considerations for using a stand-alone adapter

Consider the following items if you plan to use a stand-alone adapter:

- Stand-alone adapters have no class loader isolation.

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

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

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

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 or WebSphere Enterprise Service Bus, WebSphere Application Server Network Deployment, and WebSphere Extended Deployment support clustered environments. Clusters are groups of servers that are managed together to balance workloads and to provide high availability and scalability. When you set up a server cluster, you create a Deployment Manager profile. The HAManager, a subcomponent of the Deployment Manager, notifies the Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) container to activate the adapter instance. The JCA container provides a run time environment for adapter instances. For information about creating clustered environments, see

the following link: http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/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 handle load variations evenly. For information about the benefits of WebSphere Extended Deployment, see 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 Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) container checks the enableHASupport resource adapter property. If the value for the enableHASupport property is true, which is the default setting, all of the adapter instances are registered with the HAManager with a policy 1 of N. This policy means that only one of the adapter instances starts polling 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 about workload management, see the following link: http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/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 admin console to set the DatasourceJNDIName property for inbound processing
- Using the admin console to set the poolDataSourceJNDIName or XADataSourceJNDIName property for outbound processing

Migrating to version 7.0 of WebSphere Adapter for JDBC

By migrating to version 7.0 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 use features and capabilities present in version 7.0.

Migration considerations

WebSphere Adapter for JDBC version 7.0 may have some features and updates that might affect your existing adapter applications. Before migrating applications that use WebSphere Adapter for JDBC, you must consider some factors that might affect your existing applications.

Compatibility with earlier versions

WebSphere Adapter for JDBC version 7.0 is fully compatible with the custom business objects (XSD files) and data bindings that are created using the adapter version 6.0.2.x, version 6.1.x, and version 6.2.x and enables the existing business objects and data bindings to work well in the latest version of the adapter.

Because version 7.0 of WebSphere Adapter for JDBC is fully compatible with version 6.0.2.x, version 6.1.x, and version 6.2.x, any of your applications that used version 6.0.2.x of WebSphere Adapter for JDBC runs unchanged when you upgrade to version 7.0. However, if you want your applications to use features and functionality present in version 7.0 of the adapter, run the migration wizard.

The migration wizard replaces (upgrades) version 6.0.2.x, version 6.1.x or version 6.2.x of the adapter with version 7.0 and enables version 7.0 features and functionality for use with your applications.

Note: The migration wizard does not create components or modify existing components, such as mappers and mediators to work with version 7.0 of the adapters. If any of your applications embed an adapter that is version 6.2.x or

earlier and you are upgrading to version 7.0, and you want your applications to take advantage of the features and functions in version 7.0, you might need to change to those applications.

If the artifacts within a module have inconsistent versions, the entire module is marked as unavailable for migration and cannot be selected. Version inconsistencies are recorded in the workspace log, as they indicate that a project might be corrupted.

The adapter migration wizard in WebSphere Integration Developer version 7.0 only supports the migration of adapters from version 6.0.2.x, version 6.1.x, and version 6.2.x to version 7.0. It does not support the adapter migration from the previous versions to version 6.2.x.

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 use features and functions in version 7.0 of the adapter. When you choose to upgrade the adapter by selecting a 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.x, version 6.1.x or version 6.2.x to version 7.0, but you do not want to migrate the adapter artifacts, you can do so by deselecting the adapter artifacts from the appropriate area of the migration wizard.

Running the migration wizard without selecting any adapter artifacts installs and upgrades your adapter. As the artifacts are not migrated, your applications cannot take advantage of the features and capabilities that exist in version 7.0 of the adapter.

Migrating multiple adapters referred within a project

When a module contains one or more connector projects, each of which references to different adapters (for example, a module project that contains connector projects referring to JDBC and SAP adapters), the migration wizard identifies the artifacts belonging to each adapter and migrates these artifacts without disrupting the artifacts of other adapters.

When you select the module project and launch the migration wizard:

- The **Source connector** field lists the connector projects with the selected module project.
- The **Dependent artifact projects** area lists only the selected module project.

If you select the connector project and launch the migration wizard:

- The **Source connector** field lists only the selected connector project.
- The **Dependent artifact projects** area lists all projects which reference the selected connector project, including the module project.

Run the migration wizard in a test environment first

Because adapter migration might require you to change those applications that use version 7.0 of WebSphere Adapter for JDBC, you must 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. The following features from earlier versions of WebSphere Adapter for JDBC have been deprecated in version 7.0 and might require changes to your applications:

- Support for the DataSourceJNDIName property in outbound processing

  This property has been deprecated in version 7.0 and instead two new properties "XA DataSource JNDI name (XADataSourceJNDIName)" on page 236 and "Connection pool DataSource JNDI name (PoolDataSourceJNDIName)" on page 239 are provided as replacements. If an artifact from the earlier version is configured with the DataSourceJNDIName property, during the migration process, the new property "XA DataSource JNDI name (XADataSourceJNDIName)" on page 236 is set to the same value as the DataSourceJNDIName property. This deprecated property is still supported by the adapter at run time so that you can use adapter version 7.0 smoothly and it is displayed in the assembly editor. However, you cannot edit this property.

Performing the migration

You can migrate a project or EAR file to version 7.0 using the adapter migration wizard. When the tool is finished, the migration is complete and you can 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 is no longer compatible with previous versions of the WebSphere Process Server or WebSphere Enterprise Service Bus runtimes or WebSphere Integration Developer.

The following steps describe how to run the adapter migration wizard from the connector project menu while in the Java EE perspective in WebSphere Integration Developer.

Procedure

1. Import the PI (project interchange) file for an existing project into the workspace.

   Note: Ensure that you do not modify the contents of the RAR or copy the adapter jar file outside the connector project.
2. When projects are created in an earlier version of WebSphere Integration Developer, the Workspace Migration wizard starts automatically and selects the projects to migrate. Follow the wizard and complete the workspace migration.

For more information, see Migrating workspaces using the WebSphere Integration Developer Migration wizard.

3. Change to the Java EE perspective.
4. Right-click the module and select **Migrate connector project**.

   You can also launch the adapter migration wizard in the following ways:

   - Right-click the project in the Java EE perspective and select **Migrate adapter artifacts**.
   - From the Problems view, right-click a migration-specific message and select **Quick Fix** to correct the problem.

   Note: If the adapter type (for example, CICS/IMS adapter) is not supported by the migration wizard, the **Migrate connector project** and **Migrate adapter artifacts** menus are not available for selection. If the adapter project is of the latest version and the module projects referencing this adapter project are also of the latest version, these menus are disabled.
5. In the Select Projects window, perform the following steps:
   a. The **Source connector** field displays the name of the connector project that you are migrating. If you are migrating a module project, this field lists all the connector projects in the module project. Select the source project from the list. For more information, see Migrating multiple adapters referred within a project
   b. The **Target connector** field displays the name of the connector to which you are migrating. If you are working with more than one adapter version, this list displays the names of all the compatible connectors. Select the connector you want to migrate.
   c. The **Target version** field displays the version corresponding to the target connector that you selected in the previous step.
   d. The **Dependent artifacts project** area lists the adapter artifacts that are migrated. If you are migrating a module project, this area lists only the selected module project. If you are migrating a connector project within the module project, this area lists all projects which reference the selected connector project, including the module project. By default, all the dependent artifact projects are selected. If you do not select a dependent artifact project, that project is not migrated. You can migrate any project that you have not selected 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. For more information, see "Upgrading but not migrating a project" on page 59.
   e. Click **Next**. A warning window is displayed with the message, "Properties that are not supported in this version of the target adapter will be removed during the migration"
   f. Click **OK**.
6. In the Review Changes window, review the migration changes that occur in each of the artifacts that you are migrating. To view the details, expand each node by clicking the + sign.
7. Click **Finish**.

   Before running the migration process, the wizard performs a backup up of all projects affected by the migration. The projects are backed up to a temporary folder within the workspace. If the migration fails for any reason, or if you decide to cancel the migration before it completes, the wizard deletes the modified projects and replaces them with the projects stored in the temporary folder.

Upon completing the migration successfully, all backed up projects are deleted.

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 7.0. You do not need to run the external service wizard after exiting the adapter migration wizard.

Upgrading but not migrating a project

You can upgrade the adapter from an earlier version, to version 7.0 while choosing not to migrate the adapter project artifacts.

About this task

Running the migration wizard without selecting any adapter artifacts installs and upgrades your adapter. As the artifacts are not migrated, your applications cannot take advantage of the features and capabilities that exist in version 7.0 of the adapter.

Procedure

1. Import the PI (project interchange) file into the workspace.
2. When projects are created in an earlier version of WebSphere Integration Developer, the Workspace Migration wizard starts automatically and selects the projects to migrate. Follow the wizard and complete the workspace migration. For more information, see Migrating workspaces using the WebSphere Integration Developer Migration wizard.
3. In the Java EE perspective, right-click the project name and click **Migrate connector project**. The adapter migration wizard is displayed.
4. In the Select Projects window, clear the dependent artifact projects, and click **Next**. A warning window is displayed with the message, "The properties that are not supported in the version of the target adapter will be removed during the migration."
5. Click **OK**.
6. In the Review Changes window, review the migration changes that occur during updating the project. To view the details, expand each node by clicking the + sign.
7. Click **Finish**.

Results

The project can now be used with WebSphere Adapter for JDBC, version 7.0.

Migrating WebSphere Business Integration applications for use with Version 7.0 WebSphere Adapters

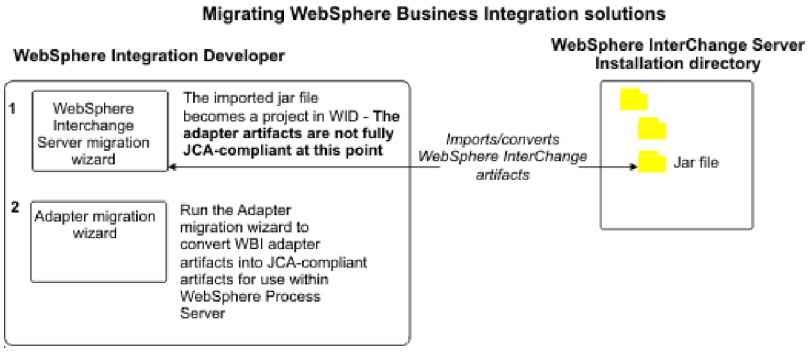
You need to migrate the WebSphere Business Integration applications so that they become compatible with Version 7.0 of your adapter.

About this task

Migrating WebSphere Business Integration applications for use with Version 7.0 of your WebSphere adapter is a multistep process. First, the artifacts from WebSphere InterChange Server are migrated and converted. A project is then created for the artifacts in WebSphere Integration Developer. In the remaining steps, the adapter-specific artifacts are migrated and converted into the JCA-compliant format supported by Version 7.0 of the adapter.

Example

The following diagram shows the wizards that you use to migrate WebSphere Business Integration solutions from WebSphere InterChange Server, so that these applications can be used with Version 7.0 of your adapter.



Roadmap for migrating applications from WebSphere InterChange Server

To use Version 7.0 of WebSphere Adapter for JDBC with applications from WebSphere InterChange Server, you need to migrate the application artifacts and convert them so that they can be deployed and run on WebSphere Process Server or WebSphere Enterprise Service Bus. Understanding this task at a high level helps you perform the steps that are needed to accomplish the task.

The following figure illustrates the flow of the migration task. 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.

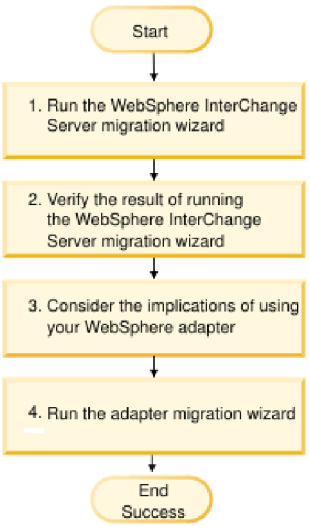


Figure 20. Roadmap for migrating applications from WebSphere InterChange Server

Migrating applications from WebSphere InterChange Server

This task consists of the following steps:

1. Run the WebSphere InterChange Server migration wizard.

   The WebSphere InterChange Server migration wizard moves the application artifacts into WebSphere Integration Developer. The migrated adapter artifacts are not fully JCA-compliant at the completion of this task.
2. Verify that the WebSphere InterChange Server migration is successful.

   Review all messages from the Migration results window and take action if required.
3. Consider the implications of using Version 7.0 of WebSphere Adapter for JDBC.

   In addition to considerations for migrating WebSphere InterChange Server applications, you need to consider how Version 7.0 of WebSphere Adapter for JDBC works with the migrated applications. Some of the adapter operations supported by WebSphere InterChange Server applications might be supported and implemented differently with Version 7.0 of the adapter.
4. Run the adapter migration wizard.

   Run the adapter migration wizard to update adapter-specific artifacts such as the schemas and service definition files (.import,.export, and .wsdl files) for use with Version 7.0 of the adapter.

Migration considerations for WebSphere Business Integration adapters

By migrating to WebSphere Adapter for JDBC Version 7.0, you have an adapter that is compliant with the Java 2 Platform, Enterprise Edition (J2EE) Connector Architecture (JCA) and designed specifically for service-oriented architecture.

Note: WebSphere Business Integration adapter and JCA-compliant adapter use different Inbound event table. You must change to the new Inbound event table supported by the JCA-compliant adapter manually after the completion of migration.

Operations considerations

Several WebSphere Business Integration adapter operations are supported differently in WebSphere Adapter for JDBC Version 7.0. You might need to provide additional development effort when you plan to use the following operations:

DeltaUpdate
The DeltaUpdate is not supported in WebSphere Adapter for JDBC. You must use the ApplyChanges operation to implement delta processing of components.

RetrieveByContent
The RetrieveAll operation is supported in WebSphere Adapter for JDBC as an equivalent to the RetrieveByContent operation.

Application artifacts

Before running the adapter migration wizard, use the WebSphere InterChange Server migration wizard to generate the application artifacts for the WebSphere Business Integration adapter, including the business objects, maps, and collaborations. Then you can run the adapter migration wizard to update the adapter-specific artifacts such as the schemas and service definition files (.import,.export, and .wsdl) so that they are suitably converted into a format that is compliant with JCA.

Run the migration wizard in a test environment first

Because migrating from a WebSphere Business Integration adapter to WebSphere Adapter for JDBC might require changes to the applications that use Version 7.0 of WebSphere Adapter for JDBC, always perform the migration in a development environment first and test your applications before deploying the application to a production environment.

Migrating application artifacts from WebSphere InterChange Server

To migrate the application artifacts into WebSphere Integration Developer, run the WebSphere InterChange Server migration wizard. The wizard imports and converts most of the artifacts into a format that is compatible with WebSphere Process Server or WebSphere Enterprise Service Bus.

Before you begin

Launch the WebSphere InterChange Server migration wizard from within WebSphere Integration Developer to migrate the application artifacts from WebSphere InterChange Server format into artifacts that are compatible with WebSphere Process Server or WebSphere Enterprise Service Bus.

For information about how to prepare to migrate artifacts from WebSphere InterChange Server and for detailed instructions on performing the migration and verifying that the migration was successful, go to the IBM WebSphere Business Process Management information center and read the topic Migrating to WebSphere Process Server from WebSphere InterChange Server.

About this task

Running WebSphere InterChange Server migration wizard might not fully convert adapter-specific artifacts (such as service descriptors, service definitions, and business objects) into WebSphere Process Server or WebSphere Enterprise Service Bus compatible artifacts. To complete the migration of adapter-specific artifacts, run the adapter migration wizard after you have successfully run the WebSphere InterChange Server migration wizard.

Note: While you run the WebSphere InterChange Server migration wizard, ensure that you set each connector in the repository to the same adapter version.

Results

The project and application artifacts are migrated and converted into WebSphere Process Sever compatible artifacts.

What to do next

Run the adapter migration wizard to migrate the adapter-specific artifacts.

Migrating adapter-specific artifacts

After a project is created for the artifacts in WebSphere Integration Developer, you can migrate the project using the adapter migration wizard. The adapter migration wizard updates adapter-specific artifacts such as the schemas and service definition files (.import, .export, and .wsdl) for use with version 7.0 of the adapter. When you finish running the adapter migration wizard, the migration is complete and you can work in the project or deploy the module.

Before you begin

Before running the adapter migration wizard, you should do the following steps:

- Review the information in the "Migration considerations" topic.
- Run the WebSphere InterChange Server migration wizard to migrate the project and convert data objects for use with WebSphere Process Server or WebSphere Enterprise Service Bus.

About this task

After migration is complete, the module will work only with Version 7.0 of your adapter.

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

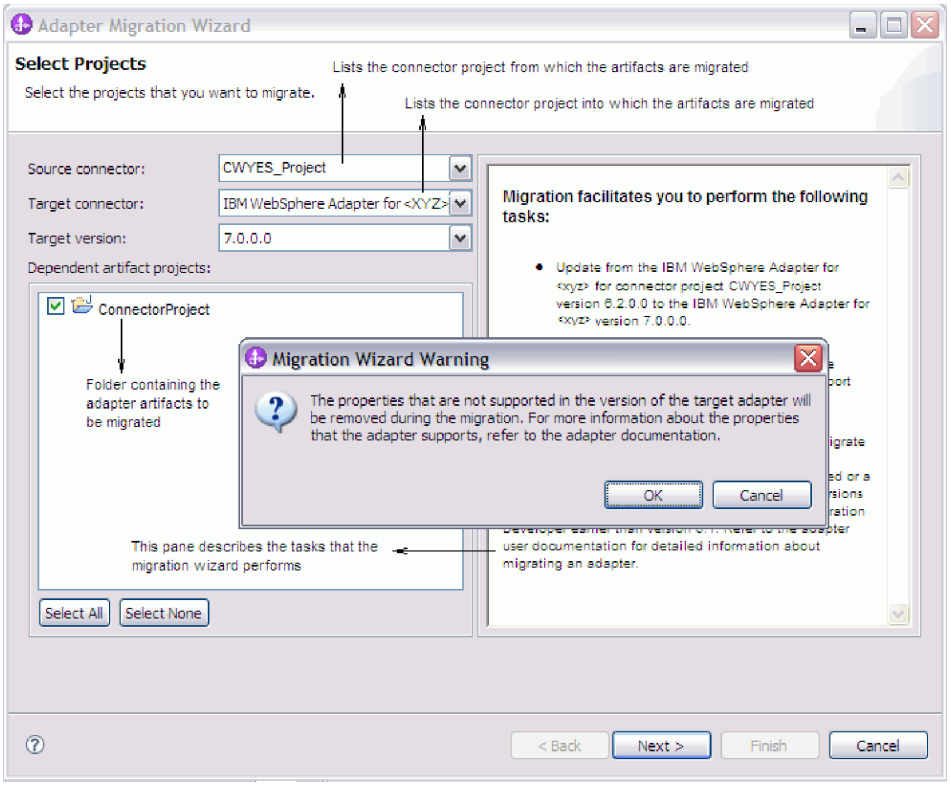
Procedure

1. Import the PI (project interchange) file for an existing project into the workspace.
2. When projects are created in an earlier version of WebSphere Integration Developer, the Workspace Migration wizard starts automatically and selects the projects to migrate. Follow the wizard and complete the workspace migration. For more information, see Migrating workspaces using the WebSphere Integration Developer Migration wizard.
3. Change to the Java EE perspective.

4. Right-click the connector project and select **Migrate connector project**.
   You can also launch the adapter migration wizard by using the right-click option and selecting the module project in the Java EE perspective and selecting **Migrate adapter artifacts**.

   Note:

   If the adapter type (for example, CICS/IMS adapter) is not supported by the migration wizard, the **Migrate connector project** and **Migrate adapter artifacts** menus are not available for selection. If the adapter project is of the latest version and the module projects referencing this adapter project are also of the latest version, these menus are disabled.
   The following figure describes the functional areas of the wizard.



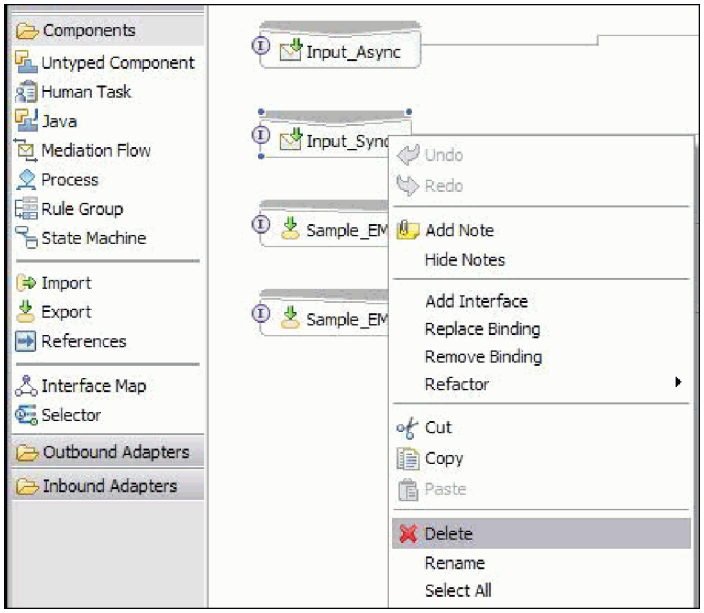
   When you launch the migration wizard from the connector project while in the Java EE perspective, by default all the dependent artifact projects are selected. If you do not select a dependent artifact project, that project is not migrated.
5. In the Select Projects window, perform the following steps:
   a. The **Source connector** field displays the name of the connector project that you are migrating. Select the source project from the list.
   b. The **Target connector** field displays the name of the connector to which you are migrating. If you are working with more than one adapter version, this list displays the names of all the compatible connectors. Select the connector to which you want to migrate.
   c. The **Target version** field displays the version corresponding to the target connector you selected in the previous step.
   d. The **Dependent artifacts project** area lists the adapter artifacts that are migrated.

e. Review the tasks and warnings presented on the welcome page, and click **Next**. A warning window is displayed with the message, "The properties that are not supported in the version of the target adapter are removed during the migration."

f. Click **OK**.

6. In the Review Changes window, review the migration changes that occur in each of the artifacts that you are migrating. To view the details, expand each node by clicking the + sign.
7. Click **Finish**.

   Before performing the migration process, the wizard backs up all projects affected by the migration. The projects are backed up to a temporary folder within the workspace. If the migration fails for any reason, or if you decide to cancel the migration before it completes, the wizard deletes the modified projects and replaces them with the projects stored in the temporary folder.
8. Select **Project > Clean**, to refresh and rebuild the workspace for the changes to take effect.
9. On successful migration, all backed up projects are deleted. Remove the Sync inbound flow manually as this flow is not used by the adapter. From the migrated project, select the Input_Sync inbound flow, right-click and select **Delete**.



10. If you are migrating an EAR file, create a new EAR file with the migrated adapter and artifacts, and deploy it to WebSphere Process Server or WebSphere Enterprise Service Bus. For information about exporting and deploying an EAR file, see "Deploying the module for production" on page 148.

Results

The project is migrated to Version 7.0. You do not need to run the external service wizard after exiting the adapter migration wizard.

Changes to the import, export, and WSDL files after migration

When the WebSphere InterChange Server migration wizard moves the application artifacts into WebSphere Integration Developer, changes made are reflected in the service definition files: the import, export and WSDL files.

The migrated adapter artifacts are not fully JCA-compliant at the completion of this task. You can complete the migration of the adapter-specific artifacts (such as service descriptors, service definitions, and business objects) to a JCA compatible format by running the adapter migration wizard.

Changes to the import file

During migration, the affected module artifacts are migrated to an import file. The existing JMS Binding property is changed to the EIS Binding property in the import file. The other property details added in the import file include information about the data binding configuration, changes to the connection information in the Managed Connection Factory properties, and several new method bindings.

Changes to the export file

During migration, the affected module artifacts are migrated to an export file. The existing JMS Binding property is changed to the EIS Binding property in the export file. The other property details added in the export file include information about the data binding configuration, changes to the connection information in the Activation Specification properties, and several new method bindings.

Changes to the WSDL file after migration

During migration, the affected module artifacts are migrated to corresponding WSDL files that include JDBC specific service description WSDL artifacts. The service description files become JCA compatible. The WSDL files will have an input and output type for each operation. Both the inbound and outbound operations work on their specific input types to produce corresponding output types after the operations are performed.

Note:

- When you migrate multiple top level inbound business objects in the project, only the first top-level business object inbound feature works correctly. For the other top level inbound business object to work correctly, you must manually modify the "emit + [verb name] + after image + [business object name]" method in the Input_Processing.java and Input_Async_Processing.java class to call the correct destination services.
- The WebSphere business integration adapter properties that are either not valid or not supported by WebSphere Adapter for JDBC are removed from the migrated artifacts.

Chapter 3. Samples and tutorials

To help you use WebSphere Adapters, samples and tutorials are available from the Business Process Management Samples and Tutorials Web site.

You can access the samples and tutorials in either of the following ways:

- From the welcome page of WebSphere Integration Developer, click **Go to Samples and Tutorials**. In the Samples and Tutorials pane, under More samples, click **Retrieve**. Browse the displayed categories to make your selection.
- From the Business Process Management Samples and Tutorials Web site: http://publib.boulder.ibm.com/bpcsamp/index.html.

© Copyright IBM Corp. 2006, 2009

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.

Road map 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 road map for the details on how to perform each of these steps.

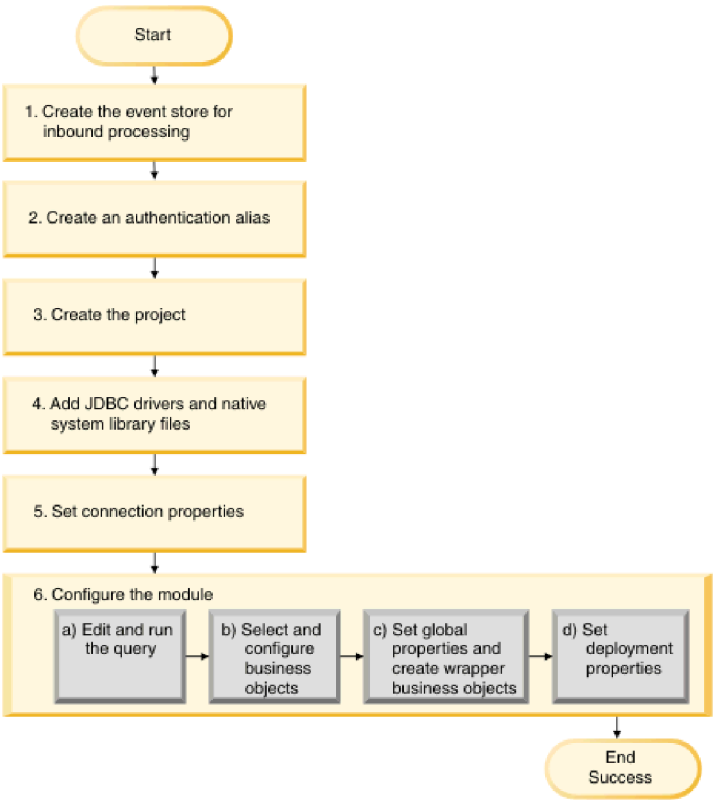


Figure 21. Road map for configuring the module

Configuring the module for deployment environment

This task consists of the following step:

1. Create the event store for inbound processing.

© Copyright IBM Corp. 2006, 2009

2. Create an authentication alias to access the database server with an encrypted password. This step is optional, depending on your policy for handling passwords and IDs. You perform this step using 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 processing 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 file 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`

   Note: Before you run the `scripts_db2_zOS.sql` script, ensure that the storage group JDDBSTO exists. For information about how to create a storage group, see CREATE STOGROUP.

   These files are located in the *WID_installation_dir*`/ResourceAdapters/JDBC_`*version*`/Scripts` directory, where *WID_installation_dir* is the installation directory for WebSphere Integration Developer, and *version* identifies the version of the adapter.

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. The adapter can use it to connect to the database instead of using a user ID and password stored in an adapter property.

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.

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 on to the administrative console and skip to step 2.

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.

To create an authentication alias, use the following procedure.

Procedure

1. Start the administrative console.

   To start the administrative console through WebSphere Integration Developer, perform the following steps:

   a. In the Business Integration perspective of WebSphere Integration Developer, 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**.

2. In the administrative console, click **Security → Global security**.
3. Under **Java Authentication and Authorization Service**, click **J2C authentication data**.
4. Create an authentication alias

   a. In the list of J2C authentication aliases that is displayed, click **New**.
   b. In the General Properties area, 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. Optional: Type a description of the alias.
e. Click **OK**.

   The newly created alias is displayed.

   The full name of the alias contains the node name and the authentication alias name you specified. For example, if you create an alias on the node widNode with the name ProductionServerAlias, then the full name is widNode/ProductionServerAlias. 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 that 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. To start the external service wizard, go to the Business Integration perspective of WebSphere Integration Developer, and then click **File → New → External Service**.
2. In the New External Service window, expand the **Adapters** node and select **JDBC**.
3. 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.
5. Click **Next**.
6. In the Import a RAR File window, accept the default project name in **Connector project** or type a different name.
7. In the **Target runtime environment** field, select the type of server where you want to deploy the module. The wizard creates the artifacts that are appropriate to that server.
8. Click **Next**. The Locate 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 external service wizard needs a copy of certain files from the database server 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

Run 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 or 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. Ensure that the JDBC driver-specific files are compatible with JRE 1.6. The following table lists the JDBC driver files needed for common database software.

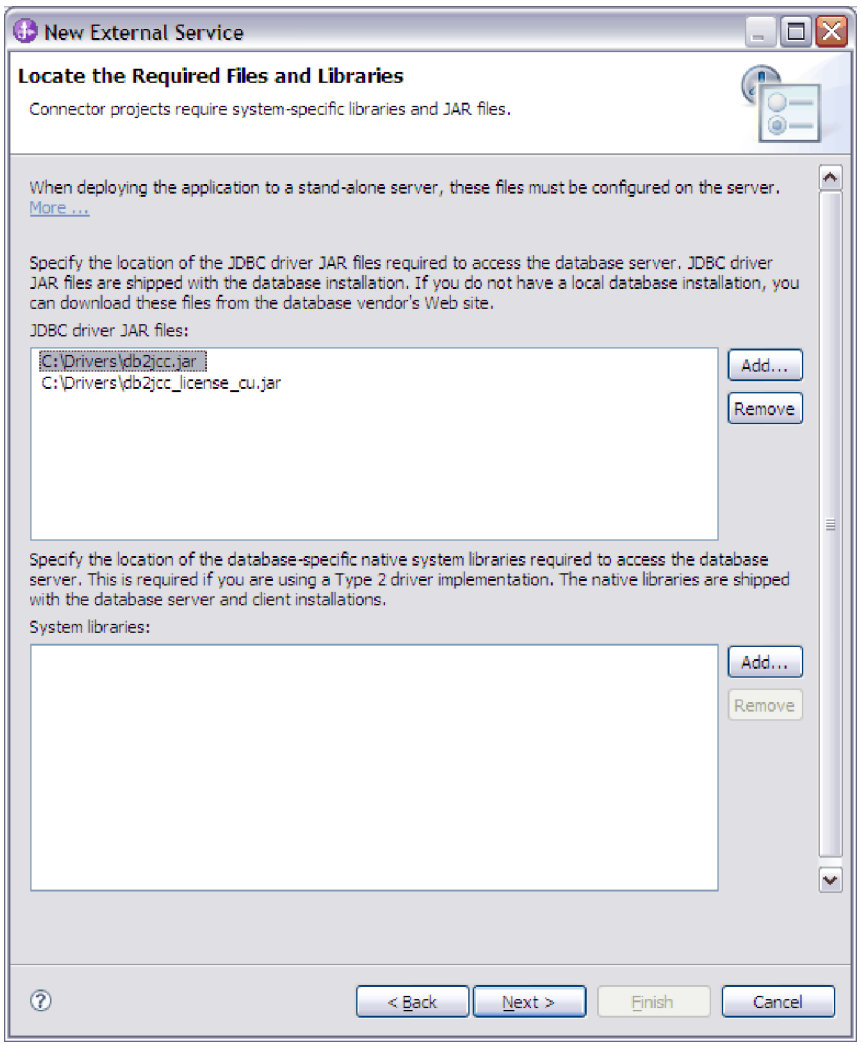
Table 11. JDBC driver files for common database software

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 (Type 4)	jt400.jar db2jcc_license_cisuz.jar	None
	IBM Toolkit for Java native driver* (Type 2)	db2_classes.jar	None
IBM DB2 Universal Database for Linux, UNIX, and Windows	IBM DB2 Universal (Type 2)	db2java.zip	None
Oracle	Thin driver	ojdbc6.jar	None

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

Database software	Driver	JDBC driver files	Native System Libraries
Microsoft SQL Server 2005	Microsoft SQL Server 2005 for JDBC	sqljdbc.jar	None
Informix	IBM Informix JDBC Driver	ifxjdbc.jar ifxjdbcx.jar plus several support .jar files For more information see, IBM Informix JDBC Driver Programmer's Guide	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 Specify the Service Generation and Deployment Properties window.			

2. In the Locate the Required Files and Libraries window, specify the location of the JDBC driver-specific files that are required by the project.
   a. In the **JDBC driver JAR files** field, click **Add** and select the JDBC driver files. For more information about JDBC drivers for see, Frequently asked questions about JDBC drivers.
   b. If you use a JDBC type 2 driver, click **Add** in **System libraries** field to add the native system libraries that are required to access the database server. If you use only JDBC type 4 drivers, leave this field empty.



3. Click **Next**. The wizard displays the Select the Processing Direction window.

Results

The wizard has the files it needs to communicate with the database server.

What to do next

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.

If you plan on discovering stored procedures in a Sybase database, be aware that the transaction mode setting on the stored procedure might potentially impact your decision on whether to use the **Set Auto Commit on database connection** property. For more information about how the transaction mode setting for stored procedures impacts your decision on whether to check **Set Auto Commit on database connection** see, *Solutions to common problems* in the Troubleshooting and support information topic.

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 213.

Procedure

1. In the Select the Processing Direction window, select **Outbound** or **Inbound**, and then click **Next**.
2. In the Specify the Discovery Properties window, specify the connection properties for the wizard to use to connect to the database. The connection properties vary depending on the database to which your adapter connects. The properties that apply to the supported databases are described in the following steps.
   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)**.

      Note: For Derby, select **Generic JDBC**.
   b. In the **JDBC driver type** field, 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.

      Note: If you are using a Derby database and you selected **Generic JDBC** as your database software, the **JDBC driver type** field pre-fills with a value of `Other`.
   c. In the **Database** field, specify the database name. For Oracle databases, this is the System ID (SID).
   d. In the **Host name** field, 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 the **Port number** field, specify the port number for connecting to the database.
   f. For DB2, Oracle, Microsoft SQL Server and Informix databases, if you select a named driver in **JDBC driver type**, the wizard provides a default value for the **JDBC driver classname** field and builds the value for the **Database URL** field from other connection fields. Additionally, for some of the named JDBC driver types, the wizard provides a default value for **Port number**. If you select the driver `Other` for any database software (for example, Derby) 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 216 and “JDBC driver class name” on page 217.

g. **For Informix database only**: In the **Server name** field, specify the name of the Informix database server to which the adapter connects.

h. In the **Additional JDBC driver connection properties** field, 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 different connection information to use at run time.

3. In the **User name** and **Password** fields, 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 at run time.

4. Optional: In the **Prefix for business object names** field, 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 the **Bidi properties** area, 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. Optional: If you are performing discovery on a Sybase database using the jConnect driver, there may be a scenario where the transaction mode setting for the stored procedure prevents the adapter from discovering result sets from the stored procedure. If the transaction mode property for the stored procedure is set to the default value of "unchained mode" or "Transact-SQL mode", select the **Set Auto Commit on database connection** check box in the **Advanced properties** area. By selecting the **Set Auto Commit on database connection** check box you bypass the "unchained mode" configuration for the stored procedure, allowing the adapter to discover result sets from the stored procedure. If you fail to select the **Set Auto Commit on database connection** check box and the transaction mode property for the stored procedure is set to "unchained mode", the adapter generates the `Cannot add object to selection` error. For more information about how the transaction mode setting for stored procedures in Sybase impact adapter processing at discovery time, see *Solutions to common problems* in the Troubleshooting and support information topic.

   Note: If the transaction mode property for the stored procedure is set to "chained mode", you do not need to select the **Set Auto Commit on database connection** check box. For a detailed property description of **Set Auto Commit on database connection**, see “Set Auto Commit on database connection” on page 219.

7. To change the location of the wizard's log files or the amount of information included in the logs, select the **Change the logging properties for the wizard** check box, and then provide the following information:

   - In the **Log file output location** field, specify the location of the log file for the wizard.

- In the **Logging level** field, specify the severity of errors that you want logged.

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

8. Click **Next**.

   If the wizard displays the window showing the exception, eccom.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 are 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 Find Objects in the Enterprise System 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 the 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:

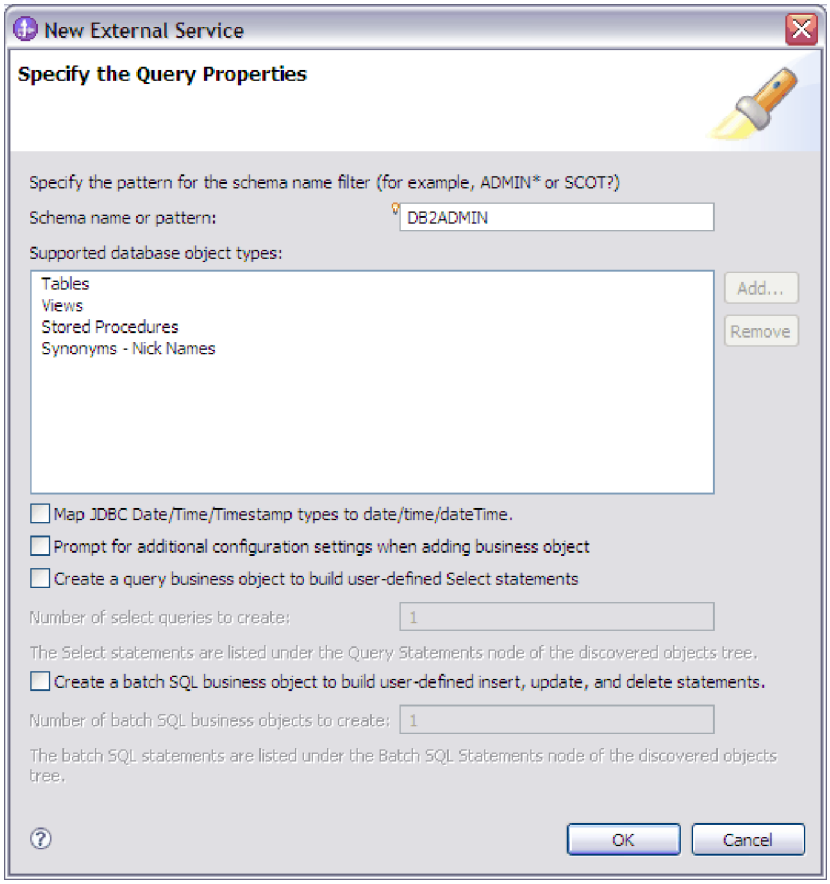
- Which schemas your module needs to access
- What type of database objects you need to access in those schemas
- Which tables, view, synonyms or nicknames, stored procedures or stored functions you need to access
- How many query and batch SQL business objects you need to create, including parameter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

About this task

This task starts in the Find Objects in the Enterprise System window of the external service wizard.

Procedure

1. In the Find Objects in the Enterprise System window, click **Edit Query**. The Specify the Query Properties window is displayed.



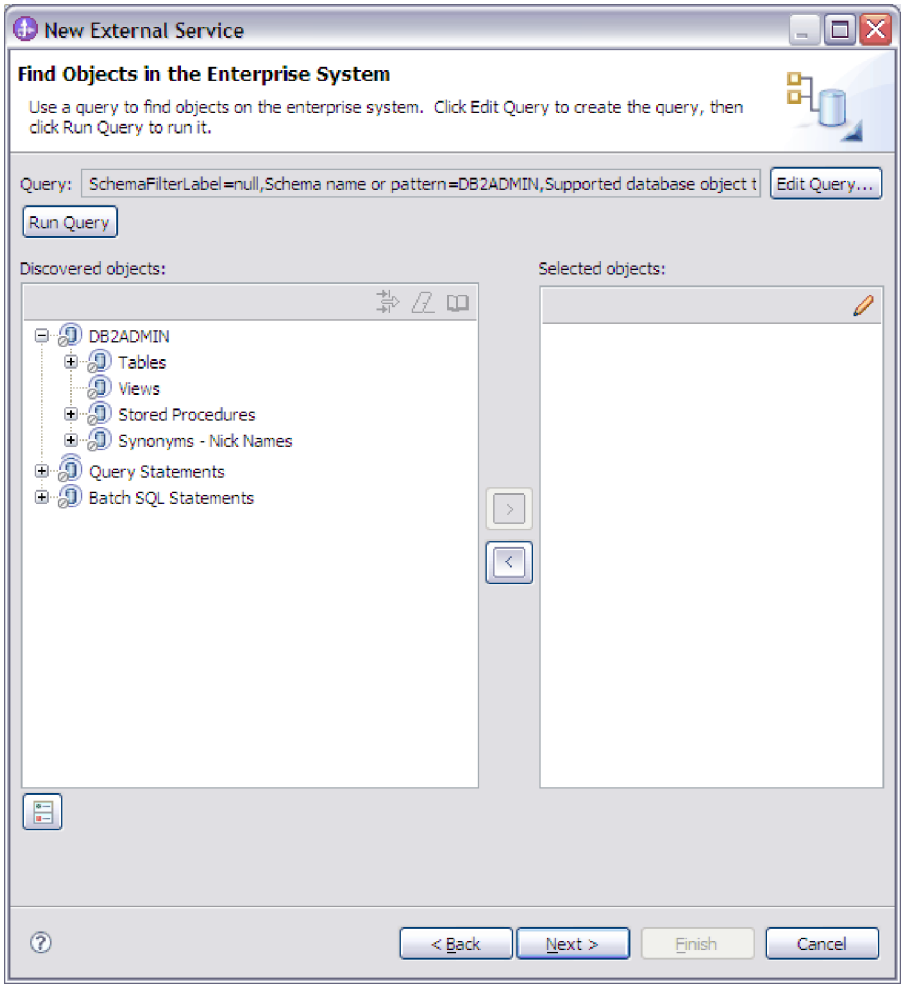
   Use the Specify the Query Properties window to perform the following tasks:
   - Reduce the search time by searching a subset of database schemas
   - Omit one or more types of database objects from the search
   - Make the wizard prompt you for application-specific information that cannot be automatically determined based on information in the database
   - Specify the number of query and batch SQL business objects you want to create
   - Map the JDBC data types Date, Time and Timestamp to date, time and dateTime

   Note: In version 6.1.x with fix pack 2, this window also allowed you to specify the number of wrapper business objects you want to create. In version 6.2.x, 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 or pattern**. Use the question mark or underscore (? or _) character to match a single character and the asterisk or percentage sign (* or %) 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 the **Supported database object types** field, and click **Remove**. To add the object type back, click **Add**. If you need to access only specific types of database objects, omitting the ones you do not need can speed up the discovery process.
4. The table, stored procedure, and stored function objects with the date, time and timestamp data types are mapped to the string data type by default. To map these objects to the actual data types that are supported by the JDBC driver such as the date, time, and datetime data types, select the **Map JDBC Date/Time/Timestamp types to date/time/dateTime** check box.
5. Select the **Prompt for additional configuration settings when adding business object** check box. Then, when 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 a hierarchy that a table business object has two attributes which are referring to attributes in two different tables (that is, it has two parent business objects), complete the configuration in the assembly editor, a tool that is 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.
6. 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.
7. 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.
8. Click **OK** to save your changes to the database query.
9. In the Find Objects in the Enterprise System 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 is shown in the following figure.



Note: To restore an expired database connection, restart the external service wizard.

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

10. 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.
11. 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 Find Objects in the Enterprise System 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. Select one or more objects in the **Discovered objects** list of the Find Objects in the Enterprise System window.
2. Click the > (Add) button.
3. The wizard opens the Specify the Configuration Properties for 'object' window.
   - If you select a single object, a single Specify the Configuration Properties for 'object' window is displayed.

     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 select multiple objects, the Specify the Configuration Properties for 'object' window displays with one page for each object selected.

     Click the name of each object in turn. The window displays the same information you would see had you selected this object individually.

     Important: Do not click **OK** until you complete the configuration pages for all the objects. The wizard does not close the notebook until you provide all the required fields, but you can close the window 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, before you start the wizard to add objects to an existing module, understand the requirements of the program that uses the business objects carefully. The wizard overwrites the existing business objects in the same path.

Selecting and configuring tables, views, and synonyms or nicknames

To select and configure business objects for tables, views, and synonyms or nicknames for use in your module, you specify the configuration properties for the business object.

Before you begin

To perform this task, you need to understand the structure of the data in the database and know what database objects the module needs to access. Specifically, you need to know the following information:

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

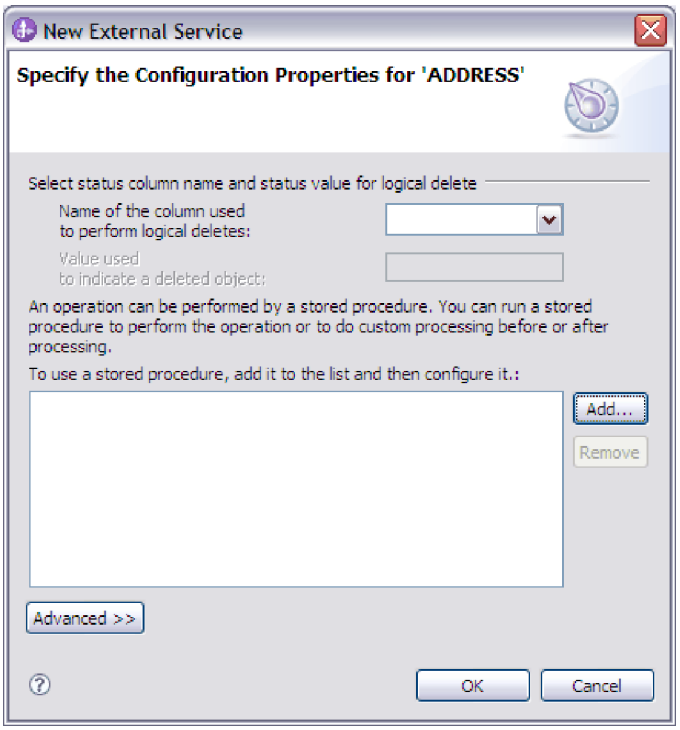
About this task

This task is performed through the external service wizard. You start in the Find Objects in the Enterprise System window and then work in a Specify the Configuration Properties for 'object' window that is specific to the business object you are configuring.

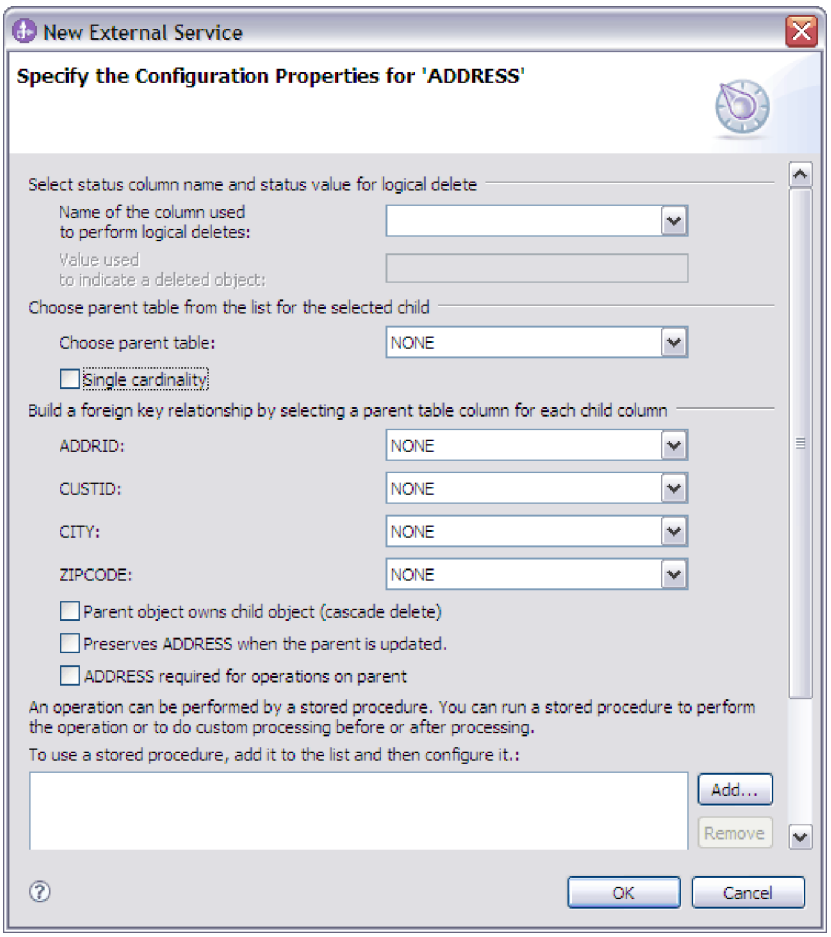
Procedure

1. In the **Discovered objects** list of the Find Objects in the Enterprise System window, select one or more tables, views, or synonyms and click the > (Add) button to add the object or objects to the **Selected objects** list.

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



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

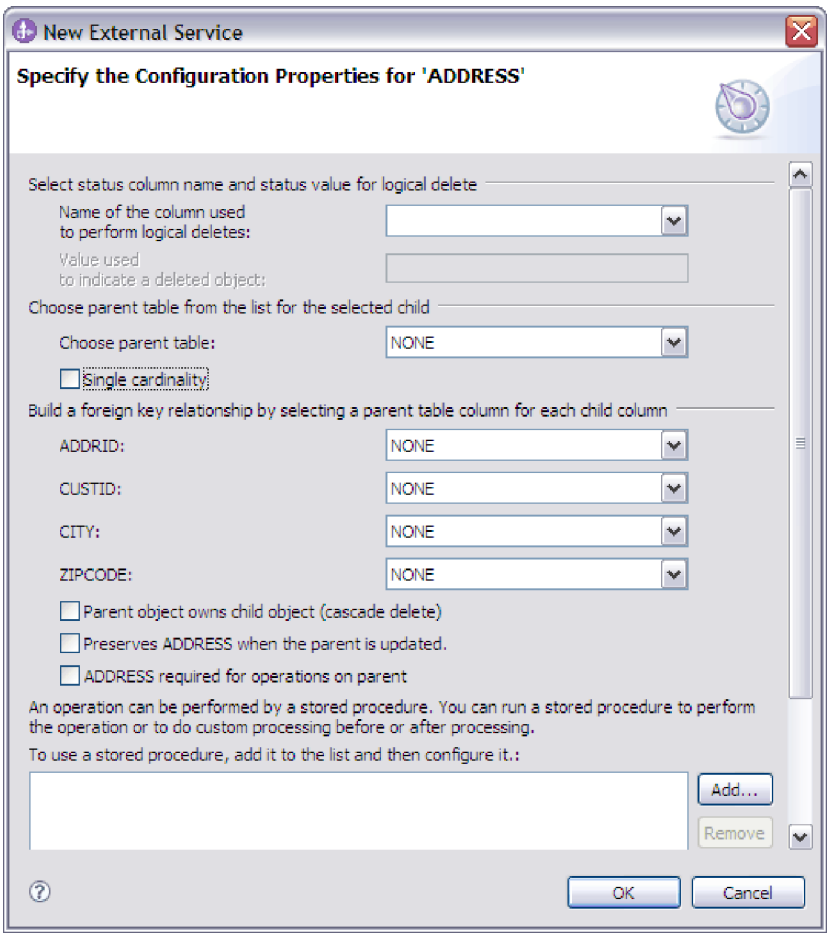


As you configure the object, choices that require advanced configuration might present additional fields in this window, causing the window to scroll. Be sure that you examine all fields on the window before clicking **OK**.

2. If the table has a column that is used to indicate logical deletes:
   a. Select the column name in the **Name of the column used to perform logical deletes** field.
   b. In the **Value used to indicate a deleted object** field, type the value that indicates that a row is logically deleted. You can get this value from your database administrator.
3. If the **Select primary key for table** *table_name* area is displayed, 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. 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.
4. Optional: Define a parent-child relationship between business objects.

   To build a parent-child hierarchy, configure the parent table first, and return to the Find Objects in the Enterprise System window to select and configure the child tables.

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



a. In the **Choose parent table** field, select the name of the parent 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 the **Single cardinality** check box. 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 has a multiple-cardinality relationship, do not select the **Single cardinality** check box. 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 exiting 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 the **Parent object owns child object (cascade delete)** check box. 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 the **Preserves *child_table_name* when the parent is updated** check box.

   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 ensure that a parent business object specifies its child business objects when the parent is submitted for a change, select the ***Child_table_name* required for operations on parent** check box.

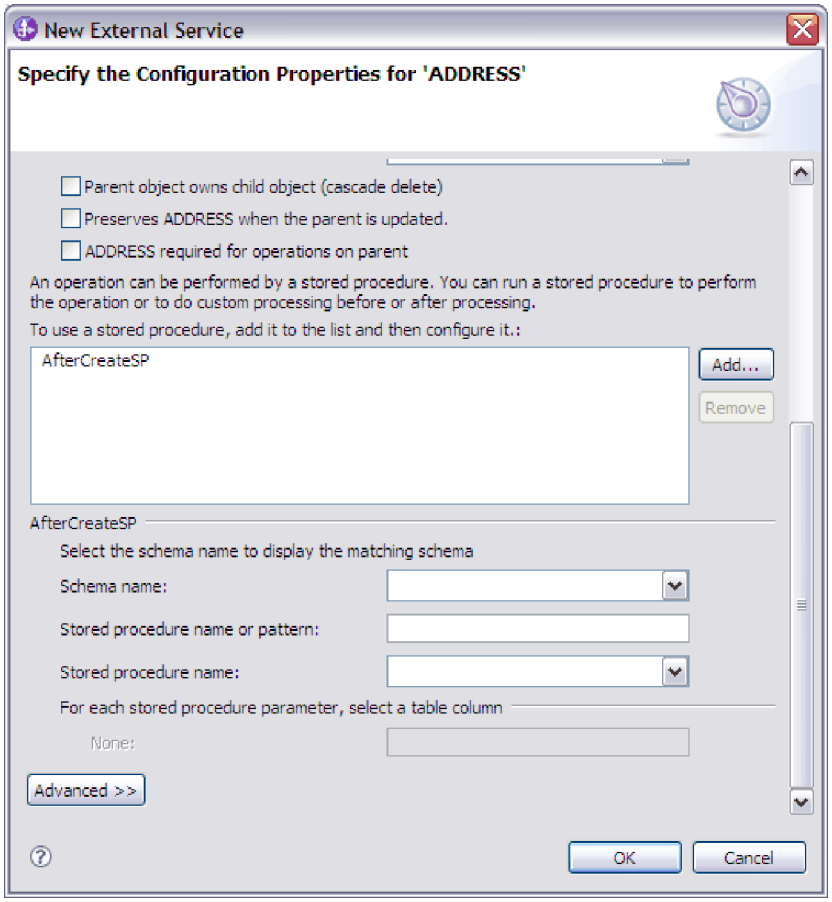
5. An operation can be performed using either a standard SQL statement generated by the adapter or using stored procedures or stored functions from the database. If you want to use stored procedures or stored functions:

   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 these stored procedures: CreateSP, BeforeCreateSP, and AfterCreateSP.

      Note: If you configure the table with RetrieveAllSP, ensure that the stored procedure returns only one result set. Set the ResultSet ASI for the stored procedure to `true` to avoid any of these exceptions being generated at run time: `No resultset found associated with the stored procedure`, `No resultset returned` or `More than one resultset returned`.

   c. Click **OK**. The Specify the Configuration Properties for 'object' 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.

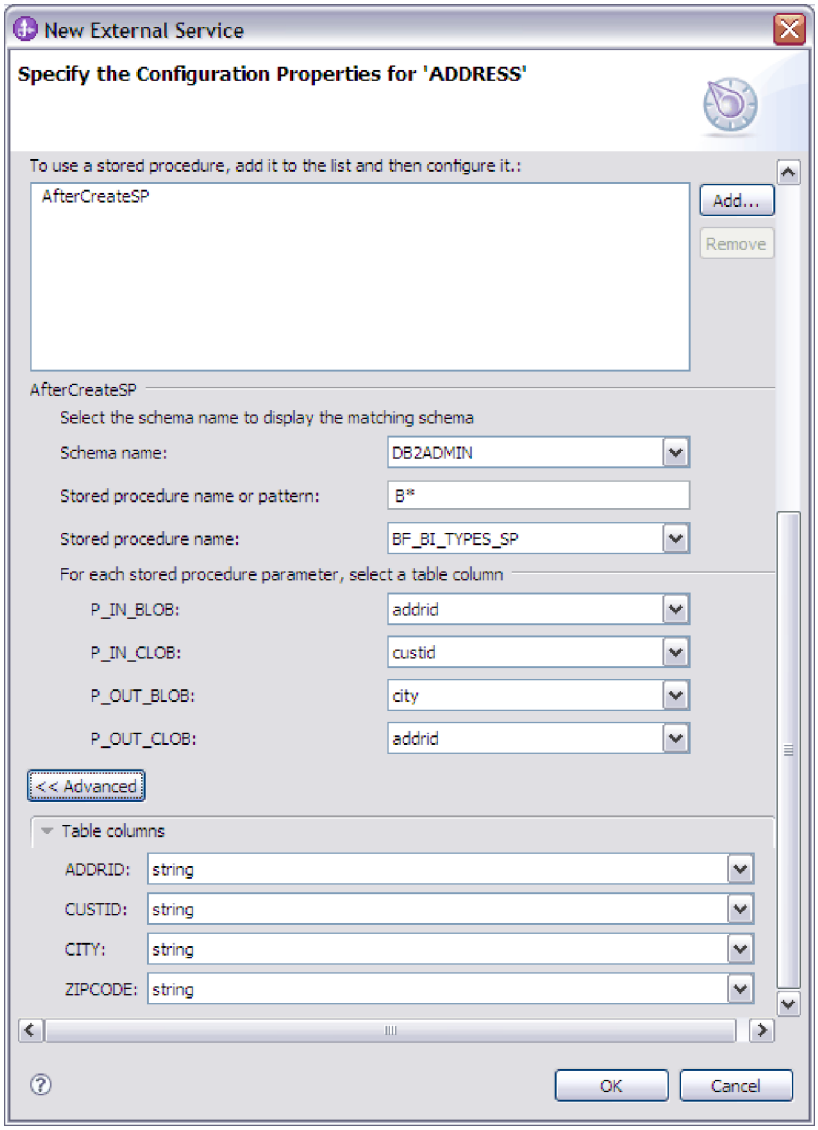


Note: In a hierarchical business object, if you want the stored procedure to be performed for each business object in the hierarchy, you must separately associate a stored procedure with the top-level business object and each child business object or array of business objects. If you associate a stored 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. In the **Schema name** field, select the name of the schema that contains the stored procedure.
   b. Specify the name of the stored procedure or stored function.
      1) In the **Stored procedure name or pattern** field, either type the name of the stored procedure or stored function, or type a name pattern. Use the question mark or underscore (? or _) to match a single character and the asterisk or percentage sign (* or %) to match multiple characters.
      2) In the **Stored procedure name** field, select the name of the procedure you want.

      The Specify the Configuration Properties for 'object' 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.



7. To specify the data type mapping for each column in the table:
   a. Click **Advanced**.
   b. Expand **Table columns**. For each column in the table, the default data type mapping is displayed.
   c. Review the mapping and make changes if required.
8. When all fields in 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 Find Objects in the Enterprise System window.
9. To change the configuration of an object from the **Selected objects** list, select the object name and then click the (Edit) icon.

What to do next

In the Find Objects in the Enterprise System window, continue to select and configure other types of business objects. When you are finished, 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 know what 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 through the external service wizard. You start in the Find Objects in the Enterprise System window and then work in a Specify the Configuration Properties for 'object' window that is specific to the business object you are configuring.

Procedure

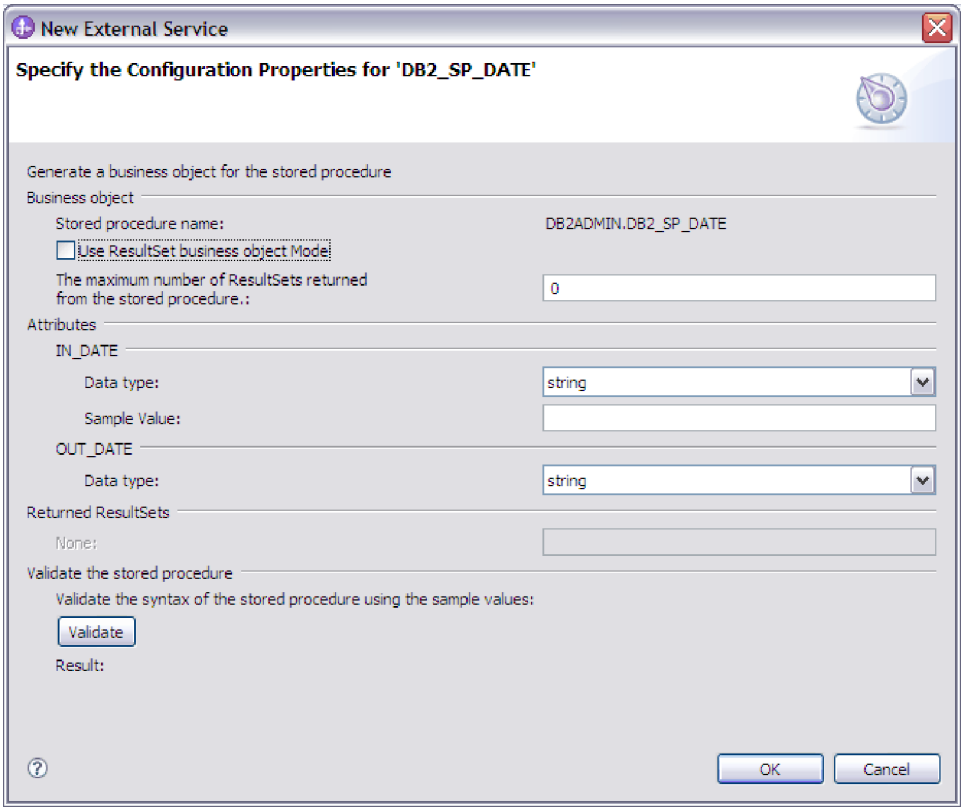
1. In the **Discovered objects** list of the Find Objects in the Enterprise System window, expand the node for the schema that contains the wanted stored procedure or stored function, and then expand the **Stored Procedures** node.
2. Filter the stored procedures by specifying a valid name or pattern for at least one of the filter fields in the Filter Properties window.
   a. Click **Stored Procedures** and then click the (Create or edit filter.) button, located at the top of the **Discovered objects** pane.
   b. In the Filter Properties window, type a name or pattern in the **Object name or pattern** field. Use the question mark or underscore (? or _) to match a single character and the asterisk or percentage (* or %) to match multiple characters. The name is case-sensitive.
   c. In the **Catalog name or pattern** field, type the name or a pattern. Use the question mark or underscore (? or _) to match a single character and the asterisk or percentage (* or %) to match multiple characters.
   d. Click **OK**.
3. Select one or more objects from the **Stored Procedures** list, and click the **>** (Add) button to add the object to the **Selected objects** list.

   Stored procedures that are defined in PL/SQL packages 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).

   Note: If the database schema has multiple stored procedures with the same name, and the stored procedures have different parameters, the external service wizard cannot identify which stored procedure is selected. Hence, the stored procedures must have unique names.

The Specify the Configuration Properties for 'object' window lists the attributes of the stored procedure business object, which include the names and data types of the parameters of the stored procedure, and information about the result sets that are returned.

Note: For DB2 database, if the table contains a structured data type, the adapter displays only the column name of the structured type. The adapter cannot obtain the type detail.



4. Select the **Use ResultSet business object Mode** check box to handle the result set returned by the stored procedure in the business object mode. With this mode, the adapter need not know the number or order of the result sets returned by the external service wizard.

   Note: If you the select the **Use ResultSet business object Mode** check box, the **The maximum number of ResultSets returned from the stored procedure** field is not available.
5. If the stored procedure returns any result set, make sure that the value for the **The maximum number of ResultSets returned from the stored procedure** field reflects the maximum number expected. The wizard creates the required 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 syntax of the stored procedure. The Oracle driver does not always return the information. If the number of returned result sets 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.
6. Configure each parameter:

a. Make sure that the **Data type** field displays the correct data type. The wizard automatically discovers the data type for a parameter that has standard JDBC data type. For other specific types, you must select the data type manually.

b. In the **Sample Value** field, type a valid value.

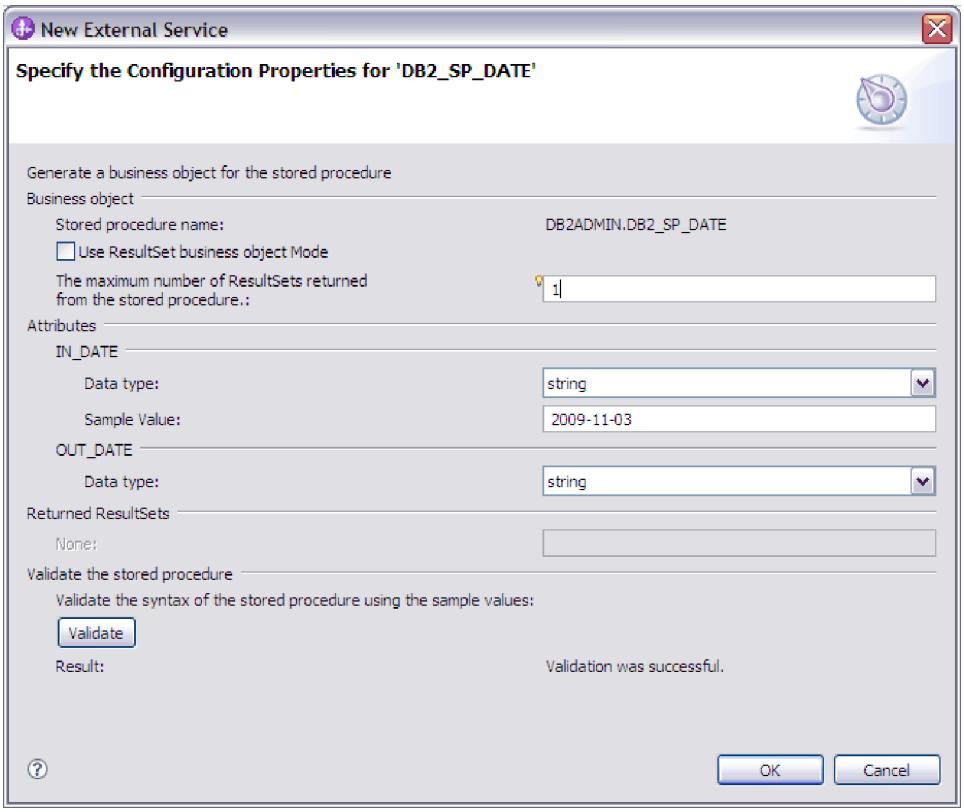
   Note: For Oracle database, if the stored procedure or stored function contains either a Varray or Object parameter with the date data type and the data type mapping is date to string, then you must specify the sample value in the yyyy-mm-dd hh:mm:ss format instead of the yyyy-mm-dd format for the stored procedure or stored function validation to be successful.

7. To validate the syntax of the stored procedure using the sample values, click **Validate**. The result of the validation is displayed in the **Result** area.

   If the **Result** area displays the `Validation failed` message, there is a problem in the information you provided. Use the error message from the database server, which follows `Validation failed` message, 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.



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 the validation succeeds. The wizard uses the results returned

during validation to create business objects to hold the result. If the stored procedure validation is not successful, the adapter does not return the result set at run time.

8. To change the configuration of an object from the **Selected objects** list, select the object name and then click the (Edit) icon.

Results

The business objects you configured for stored procedures and stored functions are listed in the Find Objects in the Enterprise System window.

What to do next

In the Find Objects in the Enterprise System window, continue to select and configure other types of business objects. When you are finished, click **Next** to set global properties and configure wrapper business objects.

Selecting and configuring 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

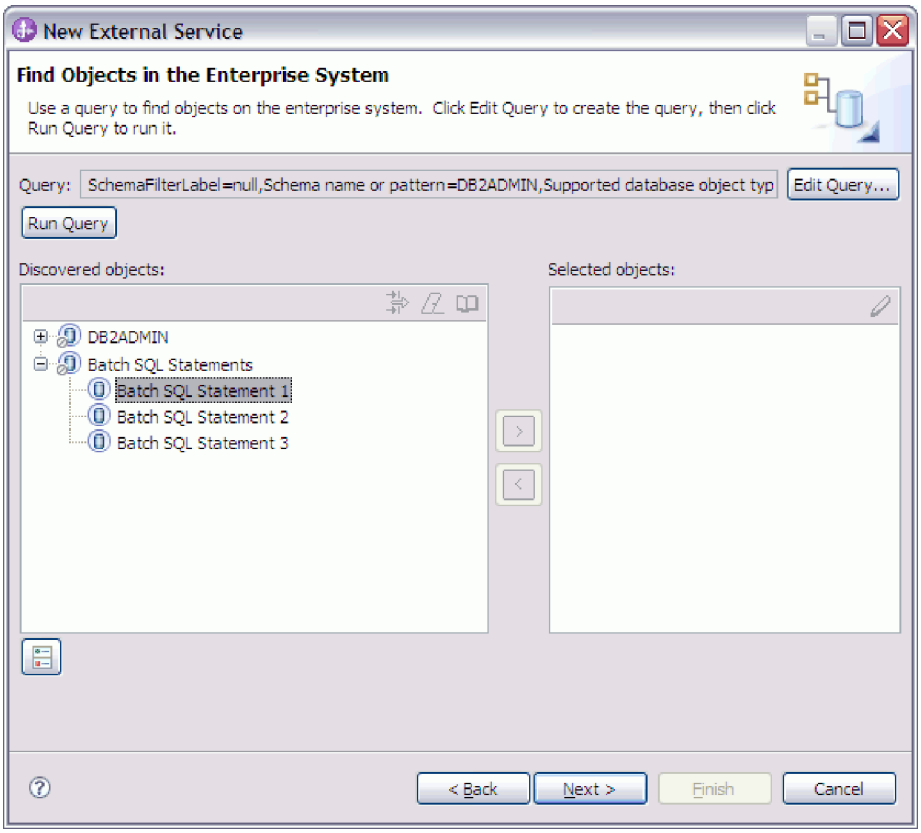
To configure batch SQL business objects, you must know the structure of the data in your database, including the tables and views. You need to know the name and data type of the columns that your SQL statements need to process. You must also be able to write SQL INSERT, UPDATE, and DELETE statements.

About this task

This task is performed through the external service wizard. You start in the Find Objects in the Enterprise System window and then work in a Specify the Configuration Properties for 'object' window that is specific to the business object you are configuring.

Procedure

1. In the **Discovered objects** list of the Find Objects in the Enterprise System 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 Specify 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.



2. Select one or more of the object templates and click the **>** (Add) button to add the object to the **Selected objects** list.
3. In the **Batch SQL business object name** field, type a name for the business object. The name cannot contain blanks, but might contain national language characters.
4. In the **SQL statements** field, 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 the **Retrieve the generated unique identifier** check box 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.

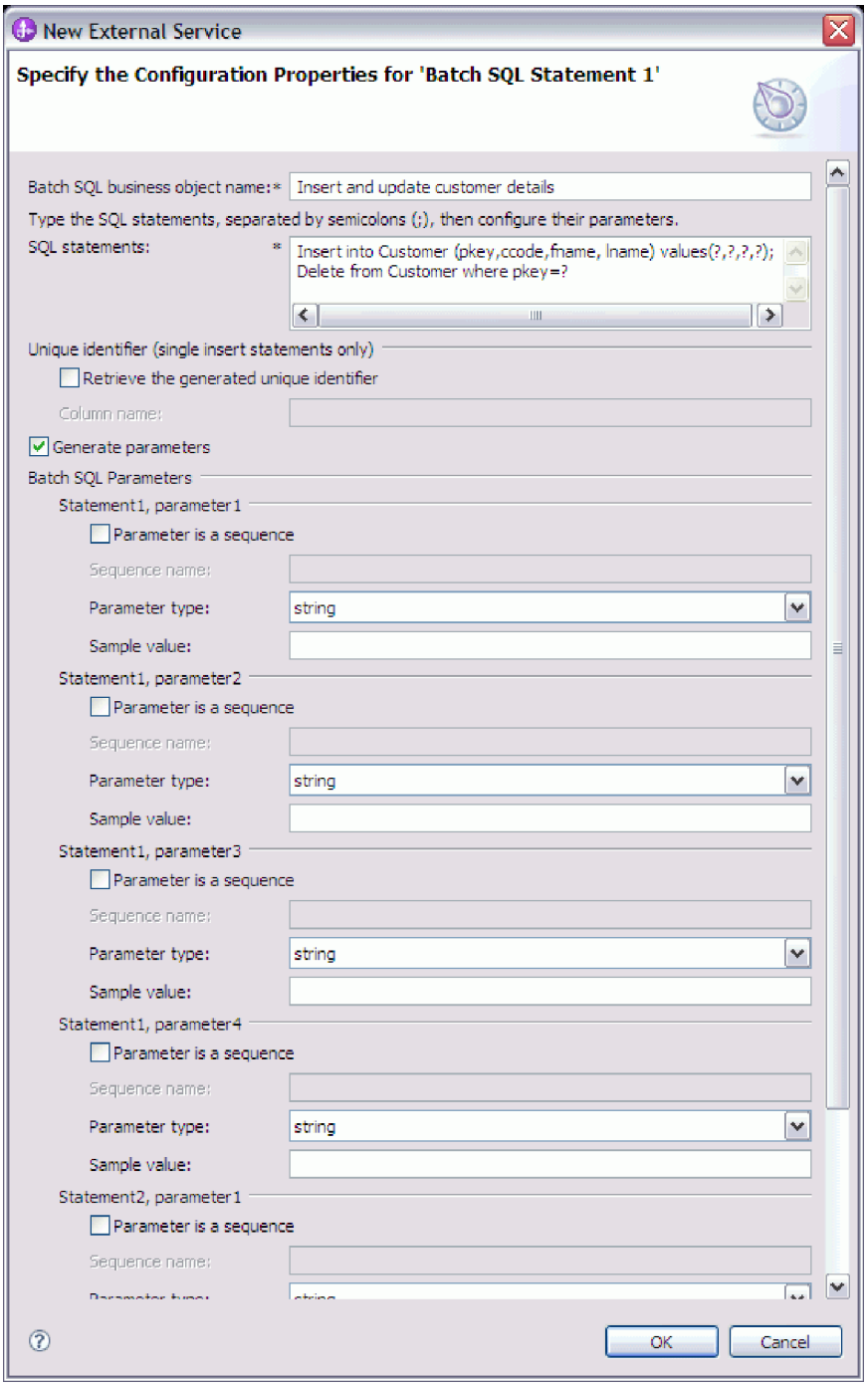
   Note: Because the Oracle database does not support using a Unique Identifier, the check box for the **Retrieve the generated unique identifier** check box is disabled if your configuration is using the Oracle database.

6. Select the **Generate parameters** check box. 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 Specify the Configuration Properties for 'object' 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 Specify the Configuration Properties for 'object' window with two SQL statements. The first statement has four parameters and the second statement has one parameter.



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. Select the **Parameter is a sequence** check box.
     b. In the **Sequence name** field, 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. Clear the **Parameter is a sequence** check box.

b. In the **Parameter type** field, select the data type of the parameter.
c. In the **Sample value** field, type a sample value of the parameter. This value is used to validate the SQL statements you entered are syntactically correct.

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

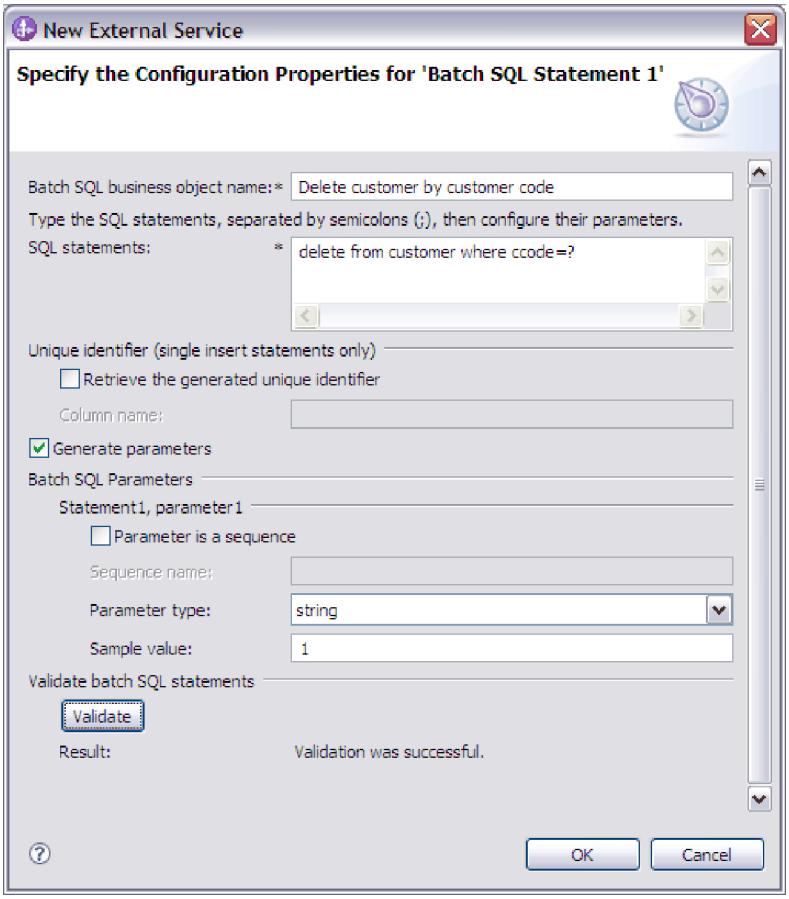
   For UPDATE and DELETE statements, you must provide a value that exists in the database. The wizard runs the statements with the sample data to get the results set, which it uses to set the attributes of the batch SQL business object. The wizard runs the statements but does not COMMIT the result, so data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

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

8. Click **Validate**. The **Result** area displays the result of the validation.

   If the **Result** area displays the `Validation failed` message, 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 that the sample data exists in the database.

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



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 Find Objects in the Enterprise System window.

What to do next

In the Find Objects in the Enterprise System window, continue to select and configure other types of business objects. When you are finished, 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

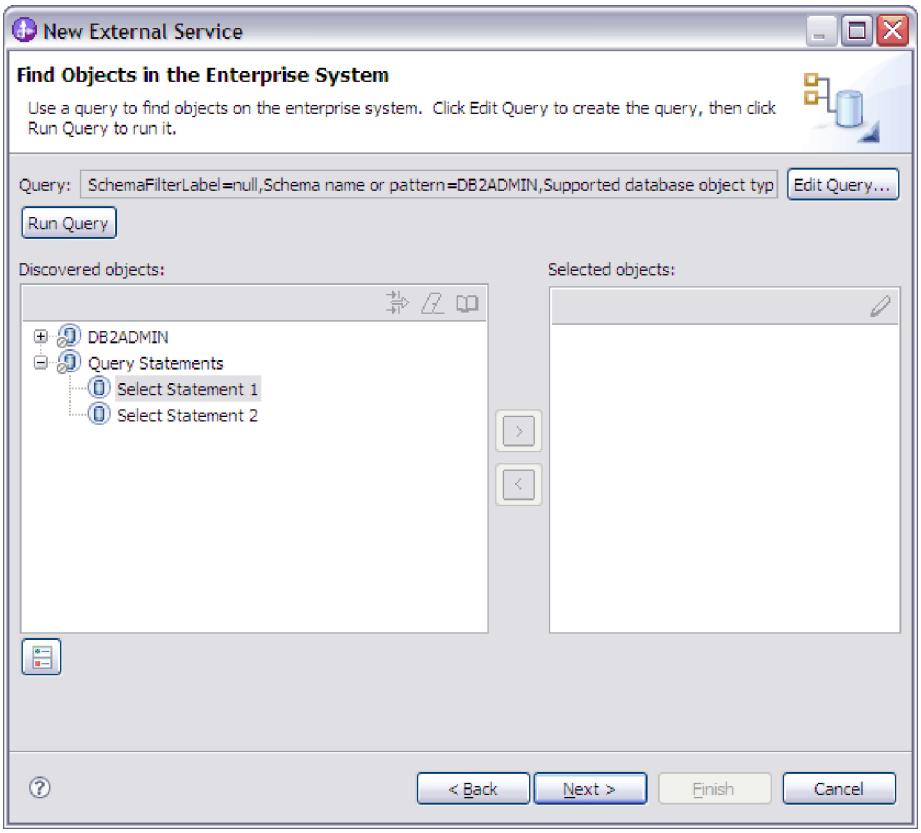
To configure query business objects, you must know the structure of the data in your database, including the tables and views. You need to know the name and data type of the columns that your module needs to access. You must also be able to write SQL SELECT statements.

About this task

This task is performed through the external service wizard. You start in the Find Objects in the Enterprise System window and then work in a Specify the Configuration Properties for 'object' window that is specific to the business object you are configuring.

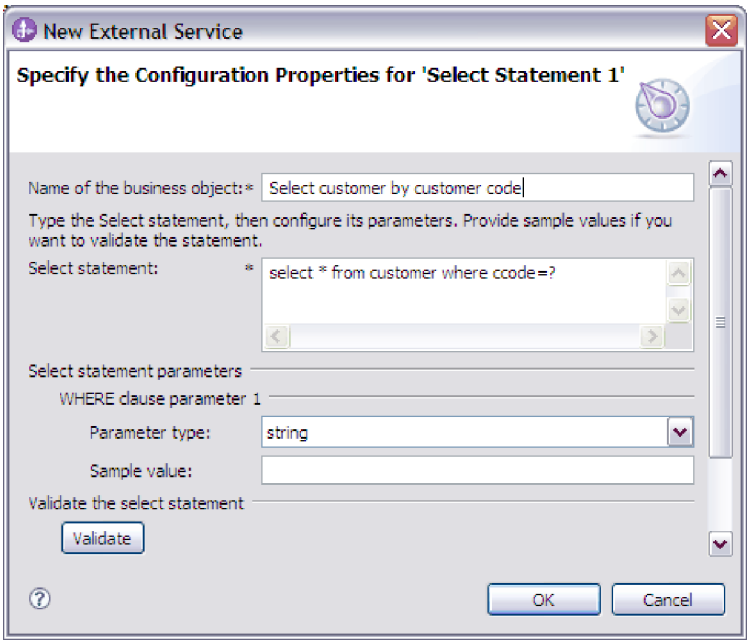
Procedure

1. In the **Discovered objects** list of the Find Objects in the Enterprise System 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 Specify 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.



2. Select one or more of the object templates and click the > (Add) button to add the objects to the **Selected objects** list.
3. In the **Name of the business object** field, type a name for the business object. The name can contain spaces and national language characters.
4. In the **Select statement** field, 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 ?)`

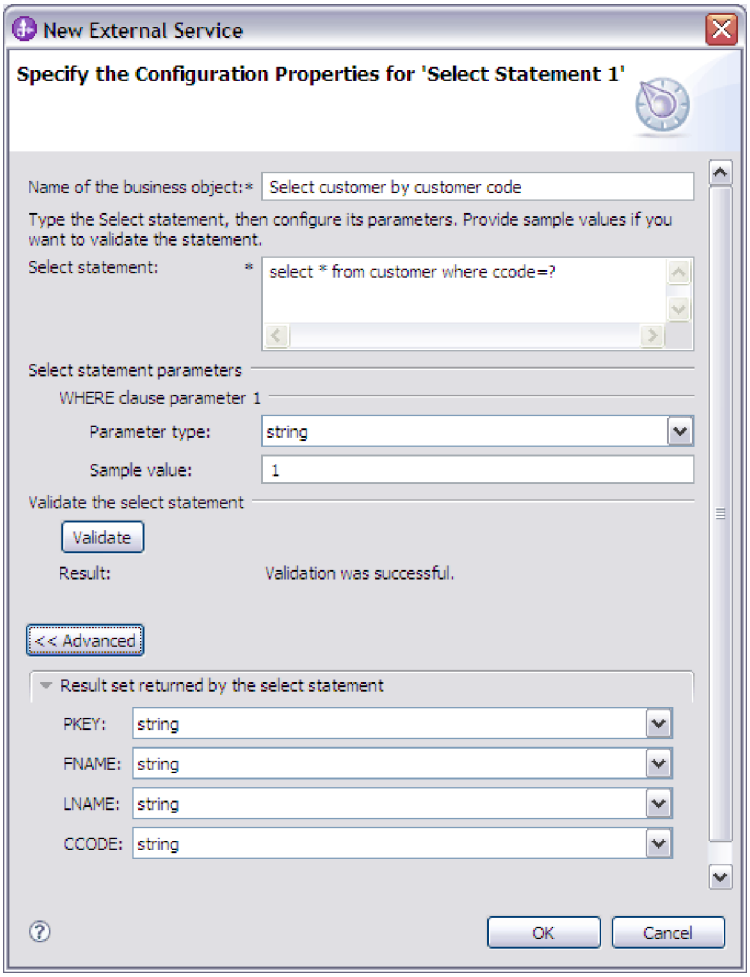
   Note: Ensure that the SELECT statement does not include a nested SELECT statement in the FROM clause.
   As you type each **?**, the window expands to display an area where you define the WHERE clause for that parameter. The following figure shows the Specify the Configuration Properties for 'object' window for a query business object that has a single parameter.



5. In the **WHERE clause parameter *n*** area, provide information about each parameter in the SELECT statement.
   a. In the **Parameter type** field, select the data type of the parameter.
   b. In the **Sample value** field, 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 **Result** area displays the result of the validation.

   If the **Result** area 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. If the validation is successful, the **Advanced** button is displayed.
7. To specify the data type mapping for each column in the result set returned by the select statement:
   a. Click **Advanced**.
   b. Expand **Result set returned by the Select statement**. For each column in the result set, the default data type mapping is displayed.
   c. Review the mapping and make changes if required.



8. Click **OK** to save the definition of the query business object.

Results

The query business objects you defined are now listed in the Find Objects in the Enterprise System window.

What to do next

In the Find Objects in the Enterprise System window, continue to select and configure other types of business objects. When you are finished, click **Next** to set global properties and configure wrapper business objects.

Setting global properties for operations and creating wrapper business objects

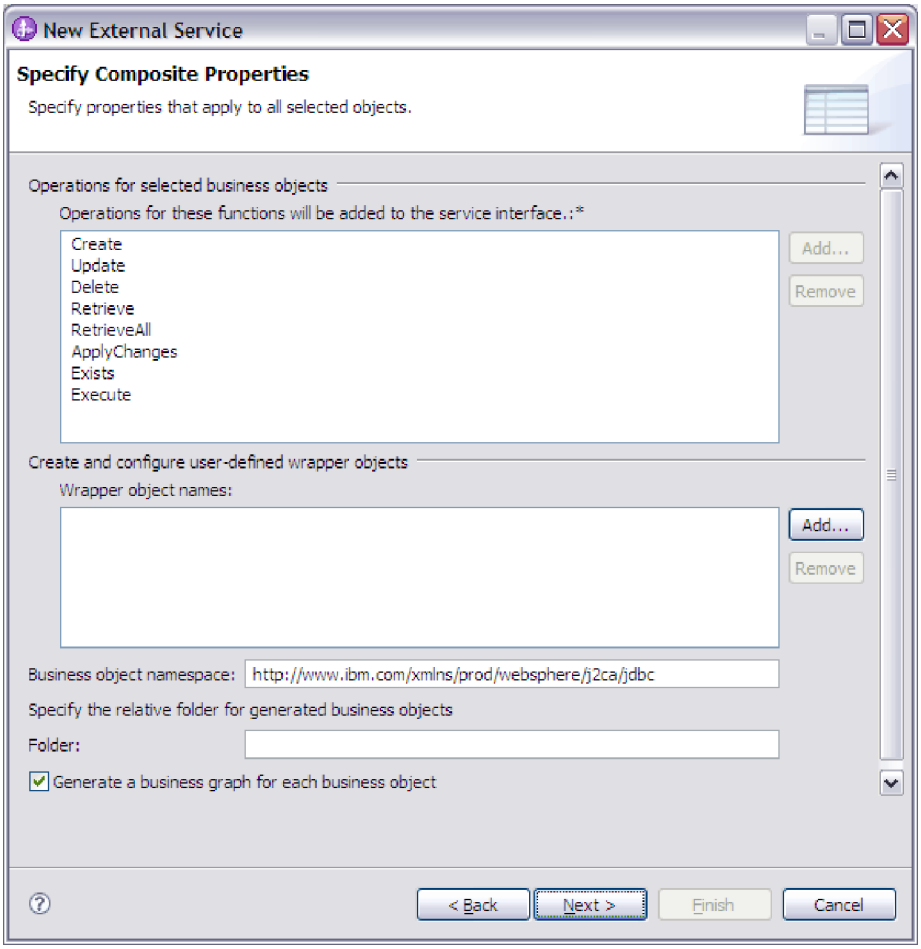
After you select database objects in the external service wizard, you need to define business objects for wrappers, and specify properties that apply to all business objects.

Procedure

1. When the **Selected objects** list in the Find Objects in the Enterprise System window contains all the business objects you want to use in your application, except for wrapper business objects, click **Next**.

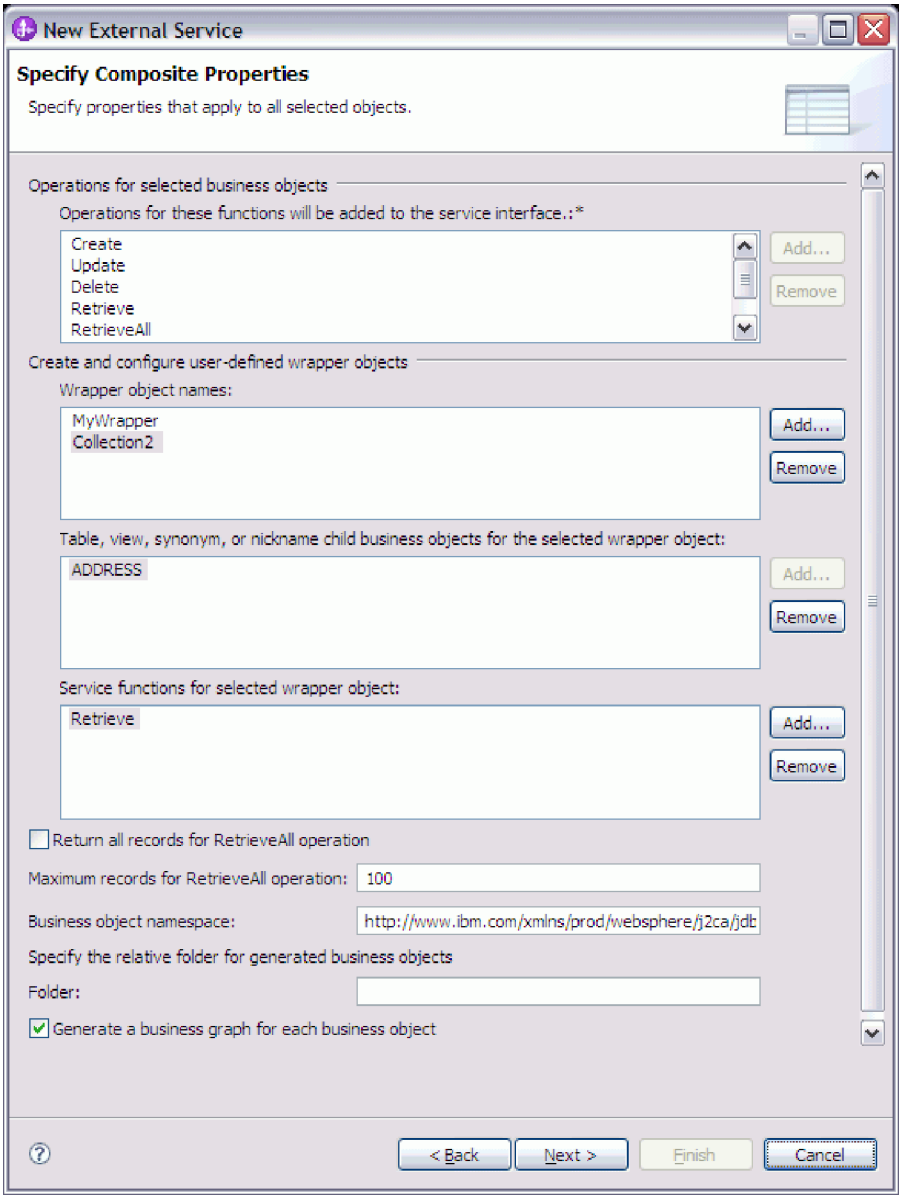
2. In the Specify Composite Properties window, review the list of operations.

   This window lists all 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.



3. To remove an operation that you do not need, select the operation name and click **Remove**. If you need to reinstate the operation, click **Add** and restore the deleted operation.
4. To create a wrapper business object:
   a. In the **Wrapper object names** area, click **Add**.
   b. In the Add Value window, type the name of your wrapper business object, and click **OK**. Do not use spaces. The name can contain national language characters.
   c. In the **Table, view, synonym, or nickname child objects for the selected wrapper object** area, click **Add**.
   d. In the Add Value window, select one or more business objects that you want to include in the wrapper, and click **OK**.
   e. In the **Service functions for selected wrapper object** area, click **Add**.
   f. In the Add Value 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 that you want to create. The following figure shows the Specify Composite Properties window with two wrapper business objects defined. The window displays the properties of one wrapper business object at a time.



5. Specify how you want to retrieve the records.
   - If you want the RetrieveAll operation to return all records matching the query, select the **Return all records for RetrieveAll operation** check box.
   - If you want to specify how many records the RetrieveAll operation must return, enter a value in the **Maximum records for RetrieveAll operation** field. The default value is 100. For more information about this property, see "Maximum records for RetrieveAll operation" on page 239.

     Note: The "Maximum records for RetrieveAll operation" on page 239 property applies only if you are using the RetrieveAll operation. This property field is disabled (unavailable for use) if you remove the RetrieveAll operation in step 3 or if you select the **Return all records for RetrieveAll operation** check box.

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

   The namespace is prefixed to the business object name to keep the business object schemas logically separated.
7. Optional: In the **Folder** field, type the relative path to the folder where the generated business objects are to be stored.

   Note: If you are creating multiple adapter artifacts within a module, ensure that you specify different business object folders for each adapter within the module. For example, if you are creating artifacts for Oracle, JDBC, SAP, and JDE within a module, you need to create different relative folders for each of these adapters. If you do not specify different relative folders, the existing artifacts are overwritten when you generate new artifacts.
8. If you want a business graph to be created for each business object, select the **Generate a business graph for each business object** check box. 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 6.2.x.

   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.

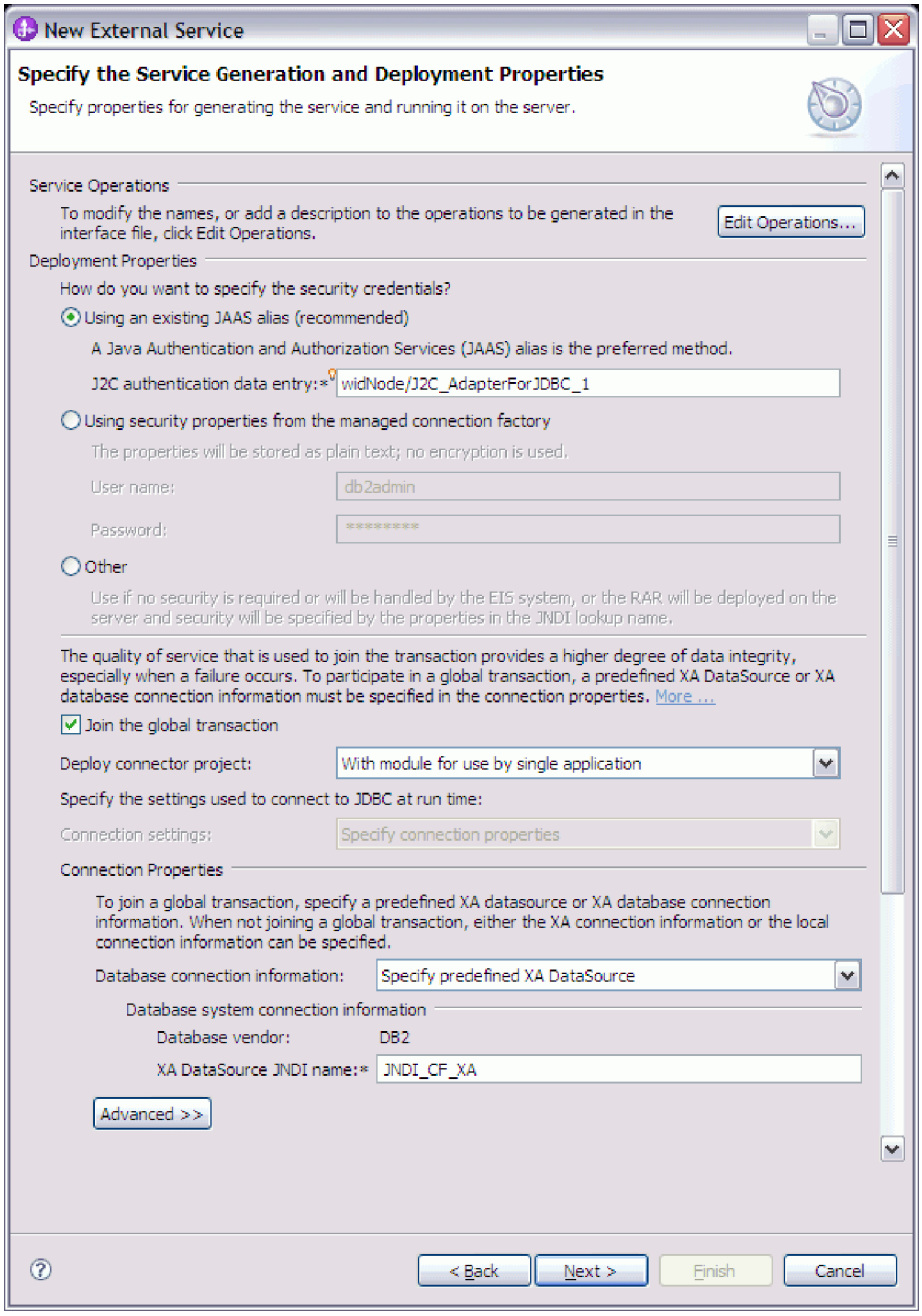
About this task

This task is performed through the Specify the Service Generation and Deployment Properties and Specify the 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 Specify the Service Generation and Deployment Properties window, click **Edit Operations** to review the names of operations or add a description for the operations for the business objects you are creating.



2. In the **Deployment Properties** area, specify how you want the adapter to get the user name and password at run time.
   - To use a J2C authentication alias, click **Using an existing JAAS alias (recommended)** and enter the name of the alias in the **J2C authentication data entry** field. 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 managed connection properties, click **Using security properties from the managed connection factory** and enter values in the **User name** and **Password** fields.
- To administer the user name and password from another mechanism, click **Other**.

Note: If you use local connection information to establish the database connection, a security credential is required. You can select either **Using an existing JAAS alias (recommended)** or **Using security properties from the managed connection factory** as the security mechanism. If you use an existing data source on the server, the security credential is not required. In this case, you can select **Other** as the security mechanism. Also, if you set the **J2C authentication data entry** field or set the **User name** and **Password** fields, they override the user name and password in the data source.

3. By default, the adapter is configured to join the global transaction. The adapter uses the XA connection for global transaction. You can configure a XA connection by specifying either a predefined XA data source or XA database connection information. For a local transaction, clear the **Join the global transaction** check box. You can configure a local transaction by specifying either a predefined connection pool data source, local database connection information, predefined XA data source or XA data source connection information.

   Note: If you select the **Join the global transaction** check box, the **Database connection information** list displays only the XA related database connection options. If an artifact from an earlier version contains only the local connection properties and the **Join the global transaction** check box was selected, after migration, the **Database connection information** list in the assembly editor displays the option for local database connection.

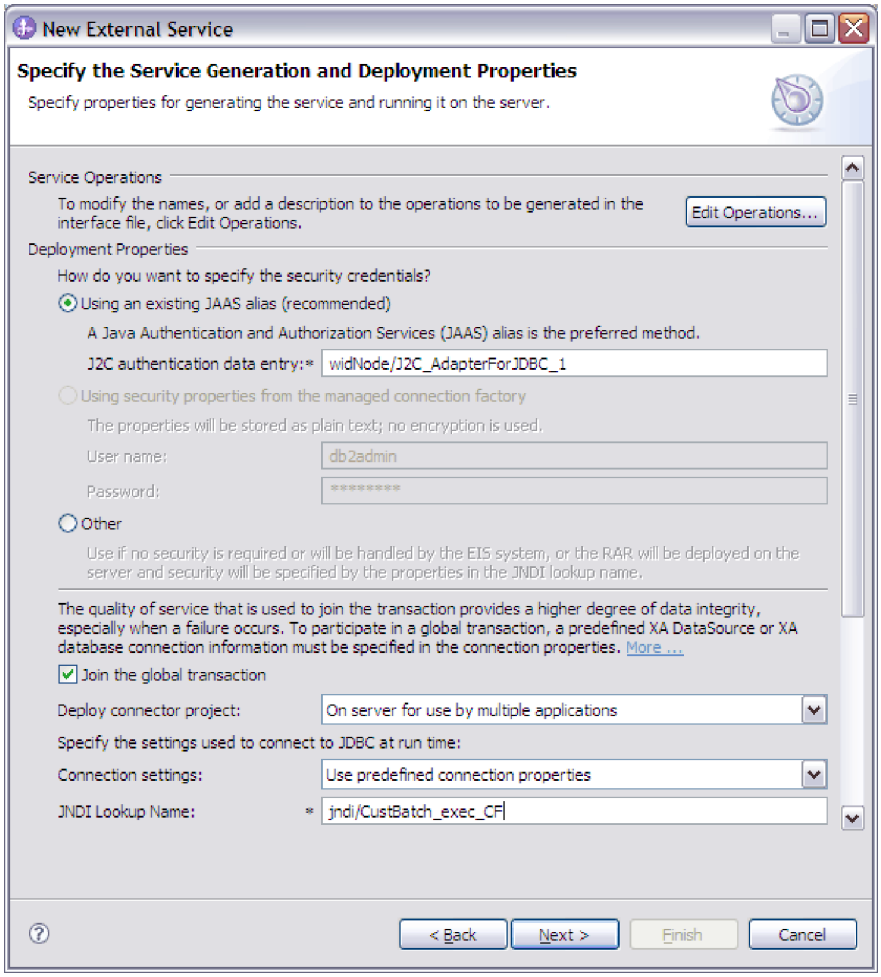
   Note: If you do not specify a valid connection information for global transaction, the adapter generates the ResourceException.
4. In the **Deploy connector project** field, specify whether to include the adapter files in the module. Select one of the following values:
   - **With module for use by single 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.
5. If you selected **On server for use by multiple applications** in the previous step, specify the connection properties used at run time.
   - 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. From the **Connection settings** list, select **Use predefined connection properties**.

b. In the **JNDI Lookup Name** field, 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.



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**.

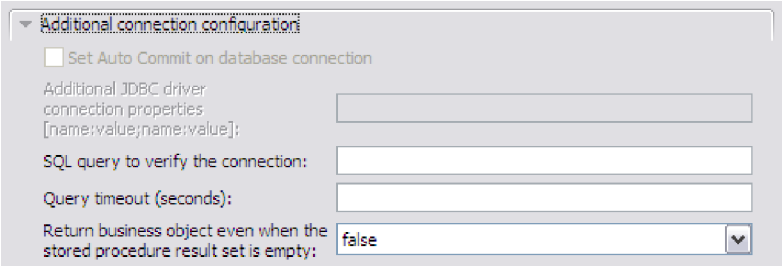
6. In the **Connection Properties** area, specify how you want the adapter to establish database connection at run time. You can establish connection with the adapter at run time in one of the following ways:
   - To use a predefined XA data source on the server (for XA connection):
     a. Select the **Join the global transaction** check box.
     b. From the **Database connection information** list, select **Specify predefined XA DataSource**.

c. In the **Database system connection information** area, enter the value in the **XA DataSource JNDI name** field. This value must be set to a JNDI data source that supports XA transactions that is created on the WebSphere Process Server or WebSphere Enterprise Service Bus. For more information about this property, see "XA DataSource JNDI name (XADataSourceJNDIName)" on page 236.

- To specify the connection information to be saved in the adapter properties (for XA connection):
  a. Select the **Join the global transaction** check box.
  b. From the **Database connection information** list, select **Specify XA database connection information**.

     Note: Using the **Specify XA database connection information** option, you can configure the adapter to support global transactions for Oracle and DB2 databases only.
  c. In the **Database system connection information** area, enter values in the **XA DataSource name** and **XA database name** fields for a DB2 database. For an Oracle database, enter values in the **XA DataSource name** and **Database URL** fields. For more information about these properties, see "Database URL (DatabaseURL)" on page 228 and "XA DataSource name (XADataSourceName)" on page 234.
- To use a predefined connection pool data source (for local connection):
  a. Clear the **Join the global transaction** check box.
  b. From the **Database connection information** list, select **Specify predefined connection pool DataSource**.
  c. In the **Database system connection information** area, enter the name of the existing JNDI data source in the **Connection pool DataSource JNDI name** field. For more information about this property, see "Connection pool DataSource JNDI name (PoolDataSourceJNDIName)" on page 239.
- To specify connection information to be saved in the adapter properties (for local connection):
  a. Clear the **Join the global transaction** check box.
  b. From the **Database connection information** list, select **Specify local database connection information**.
  c. In the **Database system connection information** area, enter values in the **Database URL** and **JDBC driver class name** fields. For more information about these properties, see "Database URL (DatabaseURL)" on page 228 and "JDBC driver class (JDBCDriverClass)" on page 230.

7. 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 is database-specific. For more information about the properties, see "Managed connection factory properties" on page 225.
8. Optional: Specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties.
   - **Additional connection configuration**



a. If you want to turn on AUTOCOMMIT for the database, select the **Set Auto Commit on database connection** check box. For more information about the property, see "Auto commit (AutoCommit)" on page 228.

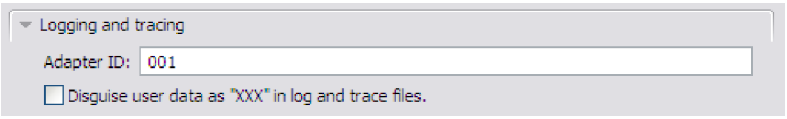
b. Set **Additional JDBC driver connection properties**. For more information about the property, see "Additional JDBC driver connection properties [name:value;name:value] (JDBCDriverConnectionProperties)" on page 227.

c. Set **SQL query to verify the connection**. For more information about the property, see "SQL query to verify the connection (PingQuery)" on page 233.

d. In the **Query timeout** field, type the time in seconds, that the adapter must wait for a response to a database query. For more information about the property, see "Query timeout (seconds) (QueryTimeOut)" on page 232.

e. Set **Return business object even when the stored procedure result set is empty**. For more information about the property, see "Return business object even when the stored procedure result set is empty (ReturnDummyBOForSP)" on page 233.

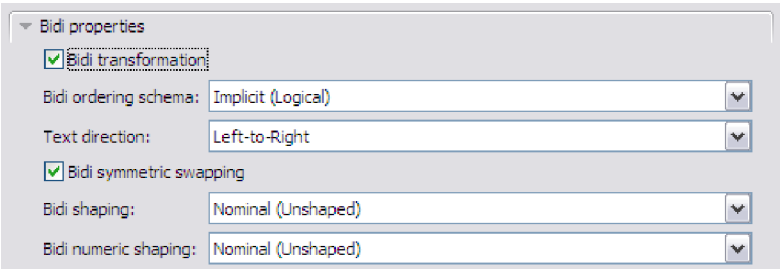
- **Logging and tracing**



– If you have multiple instances of the adapter, set **Adapter ID** to a value that is unique for this instance. For more information about the property, see "Adapter ID (AdapterID)" on page 226.

– If you want to mask certain information so that the information is not displayed in the logs or traces, select **Disguise user data as "XXX" in log and trace files**. For more information about the property, see "Disguise user data as "XXX" in log and trace files (HideConfidentialTrace) " on page 229

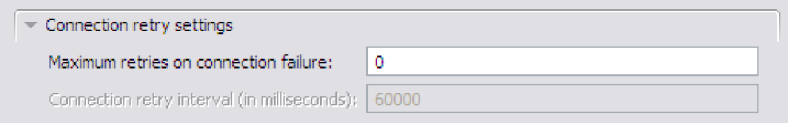
- **Bidi properties**



– To enable bidirectional support for the adapter at run time, select the **Bidi transformation** check box.

- Set the properties, ordering schema, text direction, symmetric swapping, character shaping, and numeric shaping to control how bidirectional transformation is performed.

- **Connection retry settings**



  - To specify the number of times the adapter can attempt to reconnect to the EIS in case of connection failure, set **Maximum retries on connection failure** to a value greater than or equal to zero. For more information, see "Maximum retries on connection failure (connectionRetryLimit)" on page 235.
  - To specify the time interval between retries if connection fails, set **Connection retry interval (in milliseconds)** to a value in milliseconds. This property is enabled only when the connectionRetryLimit property has a value greater than zero. For more information, see "Connection retry interval (in milliseconds) (ConnectionRetryInterval)" on page 236.

9. Click **Next**. The Specify the Location Properties window is displayed.
10. In the Specify the Location Properties window, specify the name of the module you want to create. This can be a new or existing module.
    - If the wanted 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 is replaced by the new version.
    - Otherwise, create a new module:
      a. Click **New**.
      b. In the Select a Business Integration Project Type window, select **Module** and click **Next**.
      c. In the Create a 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 the **Location** field.
      e. If you want the module to be automatically opened in the assembly diagram in WebSphere Integration Developer when the wizard closes, select the **Open the module assembly diagram** check box. Otherwise, clear this option.
      f. Click **Finish** to create the new module.
11. Specify the namespace you want to use for your artifacts.
    - If you want the business objects in the module to use the default derived namespace, select the **Use the default namespace** check box.
    - To specify a different namespace, clear the **Use the default namespace** check box and type a different value in the **Namespace** field.
12. Optional: 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.

13. In the **Name** field, accept the default import name or type a different name.
14. Optional: 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 the **Library** field.
15. Optional: In the **Description** field, type a descriptive comment about the module.
16. When you are finished setting properties, click **Finish**.

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

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

- To set the CopyAttribute parameter for a column so that its value is set to the same value as another column.
- To 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 add attributes to a business object. For example, when you run the discovery process against a table in either a DB2 or Microsoft SQL Server database, and that table defines a column as an identity column, the generated business object for that table does not include the Unique Identifier attribute. Because the adapter needs the Unique Identifier of the identity column at run time, you must add it to the attribute application-specific information. In this case, you add <UID>AUTO</UID> to the attribute application-specific information. For the oracle database, specify the sequence name to UID in order to define a field as an auto-generated field since Oracle does not support identity column.

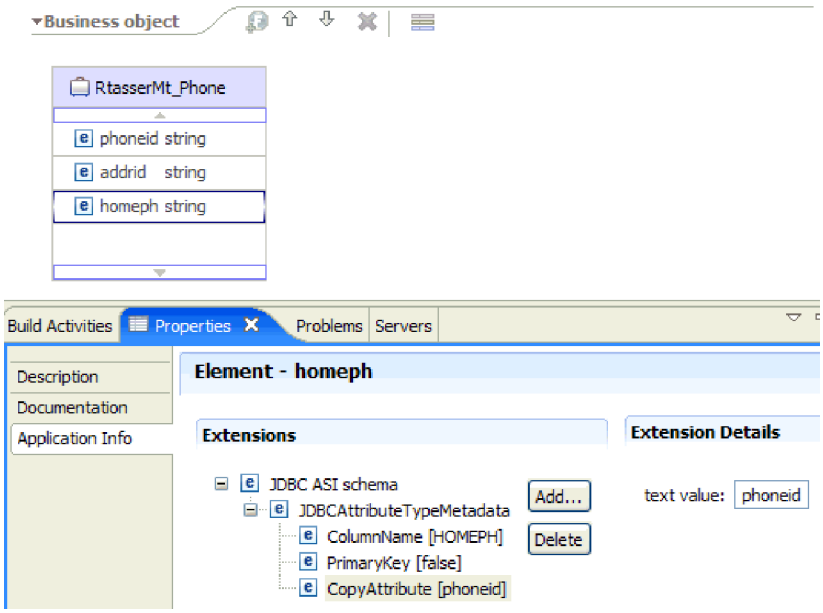
  Note: You do not need to make similar changes to a table business object if you generated it from an Informix database. When you run the discovery process against a table in an Informix database, and that table defines a column as serial (identity columns are referred to as *serial columns* in Informix), the resulting business object does include the Unique Identifier attribute for the serial column, and therefore you need not edit the business object application-specific information. The generated value of the Unique Identifier parameter for a serial column in an Informix database table is either `serial` or `serial8`.
- 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, such as the changes mentioned above, 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 `e-mail` column, set the CopyAttribute parameter of the `contact` attribute to `e-mail`. 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 **CopyAttribute** property.
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 with the CopyAttribute property set to a column in the current table.



Results

The business object is configured to use the CopyAttribute property to set the business object attribute and properties for a database column based on information in another 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:

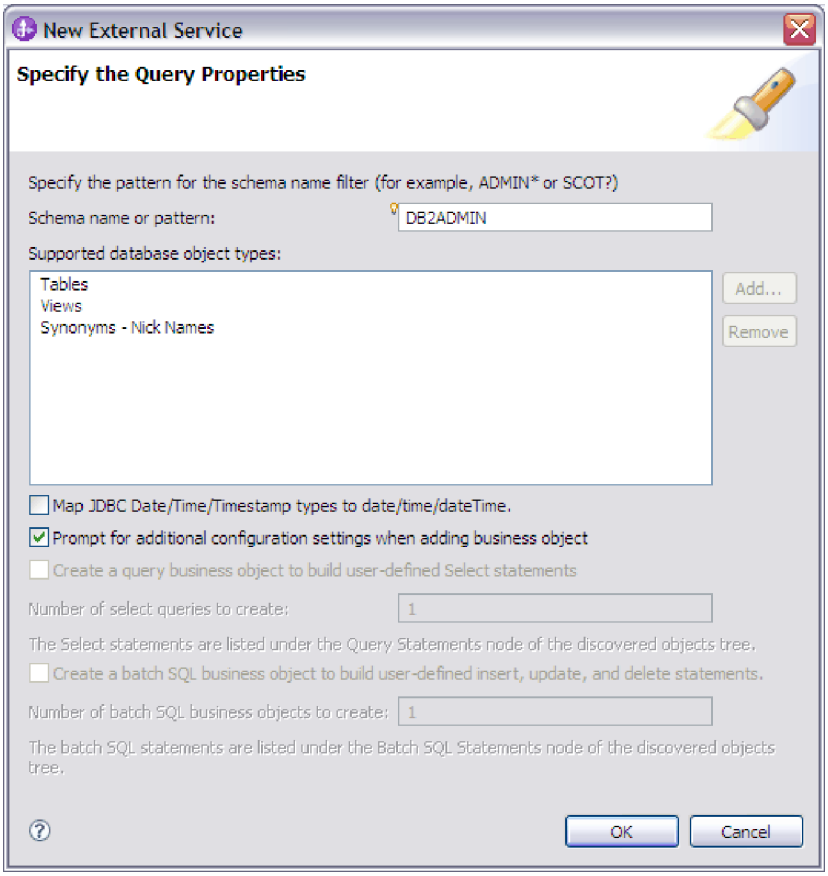
- Which schemas your module needs to access
- What type of database objects you need to access in those schemas

About this task

This task starts in the Find Objects in the Enterprise System window of the external service wizard.

Procedure

1. In the Find Objects in the Enterprise System window, click **Edit Query**. The Specify the Query Properties window is displayed.



Note: The 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** are available only for outbound processing.

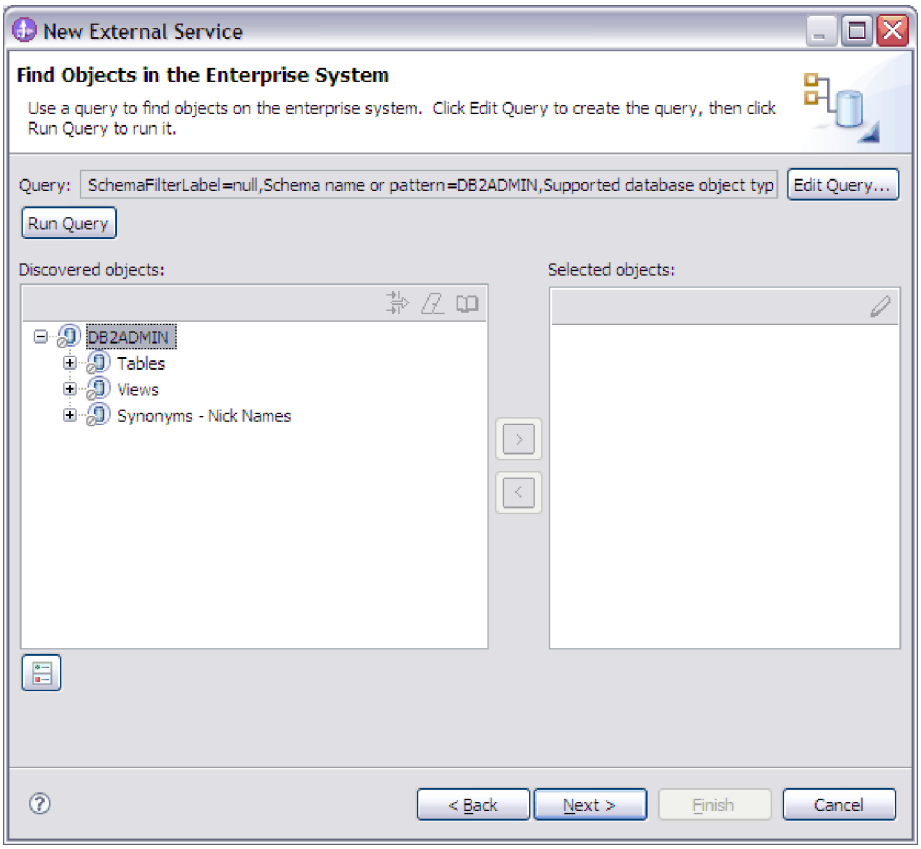
Use the Specify the Query Properties window to perform the following tasks:

- 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
- Map the JDBC data types Date, Time and Timestamp to date, time and dateTime

2. To limit the number of database schemas that are retrieved, type the name of the schema or a name pattern in **Schema name or pattern**. Use the question mark or underscore (? or _) character to match a single character and the asterisk or percentage sign (* or %) 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 the **Supported database object types** field, and click **Remove**. To add the object type back, click **Add**. If your database contains object types that you do not need to access, omitting them can speed up the discovery process.

4. Table objects with the date, time and timestamp data types are mapped to the string data type by default. To map these objects to the actual data types that are supported by the JDBC driver such as the date, time, and datetime data types, select the **Map JDBC Date/Time/Timestamp types to date/time/dateTime** check box.
5. Select the **Prompt for additional configuration settings when adding business object** check box. Then, when 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 a hierarchy that a table business object has two attributes which are referring to attributes in two different tables (that is, it has two parent business objects), complete the configuration in the assembly editor, a tool that is 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.
6. Click **OK** to save your changes to the query.
7. In the Find Objects in the Enterprise System window, click **Run Query** to use the query to discover database objects. The result of running a typical query is shown in the following figure.



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

8. 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 Find Objects in the Enterprise System 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. Select one or more objects in the **Discovered objects** list of the Find Objects in the Enterprise System window.
2. Click the **>** (Add) button.
3. The wizard opens the Specify the Configuration Properties for 'object' window.
   - If you select a single object, a single Specify the Configuration Properties for 'object' window is displayed.

     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 select multiple objects, the Specify the Configuration Properties for 'object' window displays with one page for each object selected.

     Click the name of each object in turn. The window displays the same information you would see had you selected this object individually.

     Important: Do not click **OK** until you complete the configuration pages for all the objects. The wizard does not close the notebook until you provide all the required fields, but you can close the window 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, before you start the wizard to add objects to an existing module, understand the requirements of the program that uses the business objects carefully. The wizard overwrites the existing business objects in the same path.

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 perform this task, you need to understand the structure of the data in the database and know what database objects the module needs to access. Specifically, you need to know the following information:

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

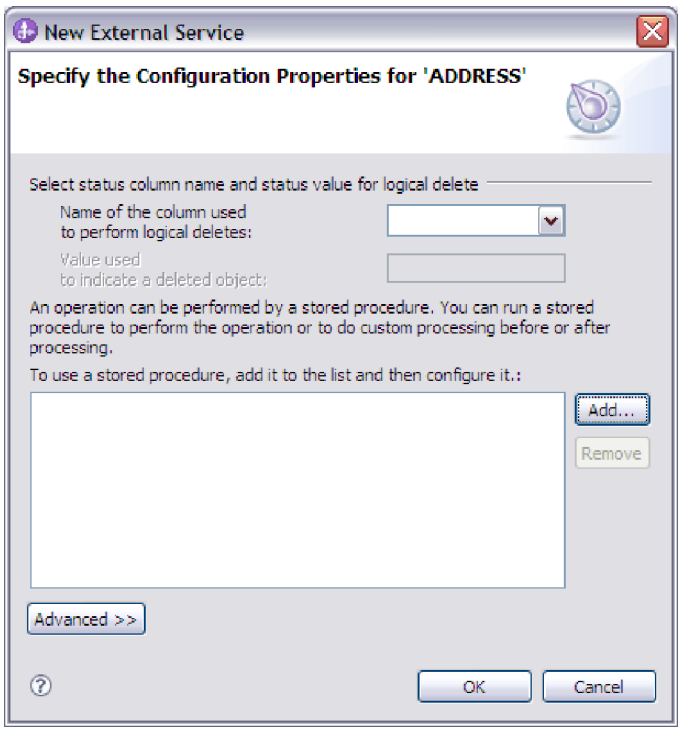
About this task

This task is performed through the external service wizard. You start in the Find Objects in the Enterprise System window and then work in a Specify the Configuration Properties for 'object' window that is specific to the business object you are configuring.

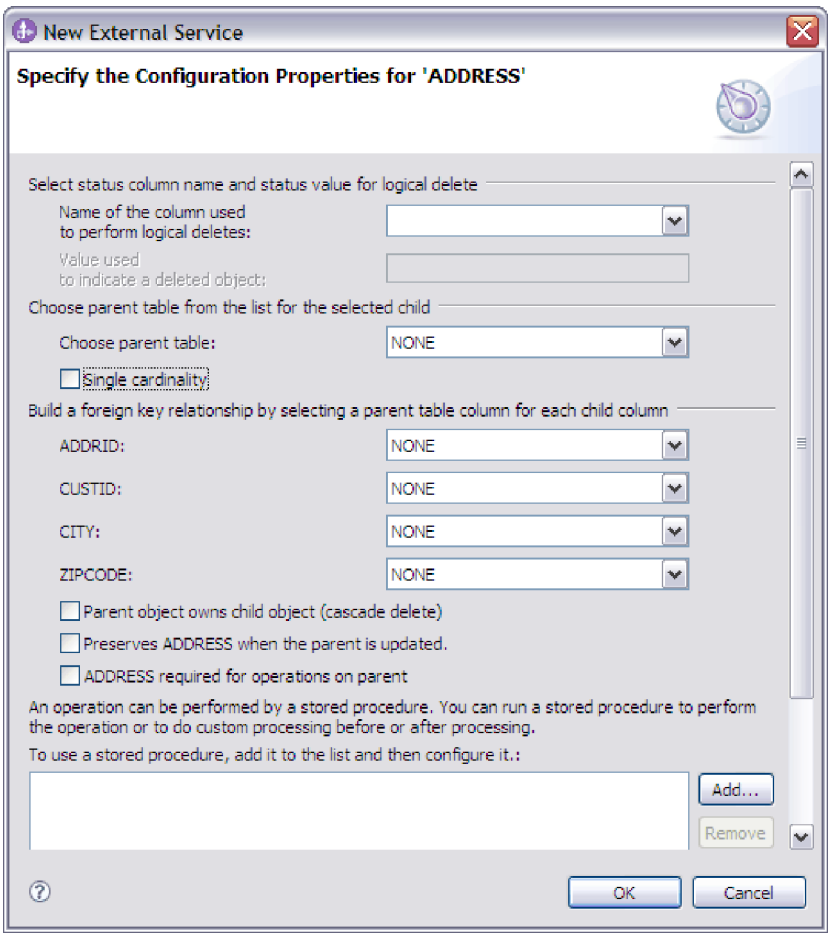
Procedure

1. In the **Discovered objects** list of the Find Objects in the Enterprise System window, select one or more tables, views, or synonyms and click the > (Add) button to add the object or objects to the **Selected objects** list.

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



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

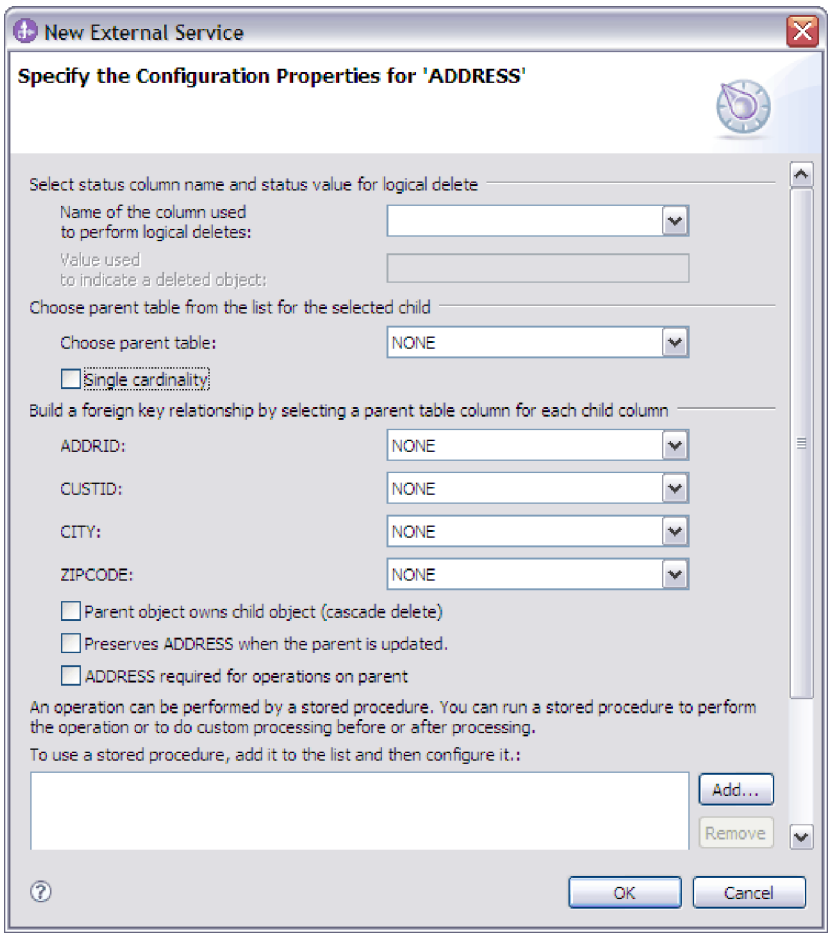


As you configure the object, choices that require advanced configuration might present additional fields in this window, causing the window to scroll. Be sure that you examine all fields on the window before clicking **OK**.

2. If the table has a column that is used to indicate logical deletes:
   a. Select the column name in the **Name of the column used to perform logical deletes** field.
   b. In the **Value used to indicate a deleted object** field, type the value that indicates that a row is logically deleted. You can get this value from your database administrator.
3. If the **Select primary key for table** *table_name* area is displayed, 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. 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.
4. Optional: Define a parent-child relationship between business objects.

   To build a parent-child hierarchy, configure the parent table first, and return to the Find Objects in the Enterprise System window to select and configure the child tables.

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



a. In the **Choose parent table** field, select the name of the parent 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 the **Single cardinality** check box. 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 has a multiple-cardinality relationship, do not select the **Single cardinality** check box. 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 exiting 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 the **Parent object owns child object (cascade delete)** check box. 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 the **Preserves *child_table_name* when the parent is updated** check box.

   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 ensure that a parent business object specifies its child business objects when the parent is submitted for a change, select the ***Child_table_name* required for operations on parent** check box.

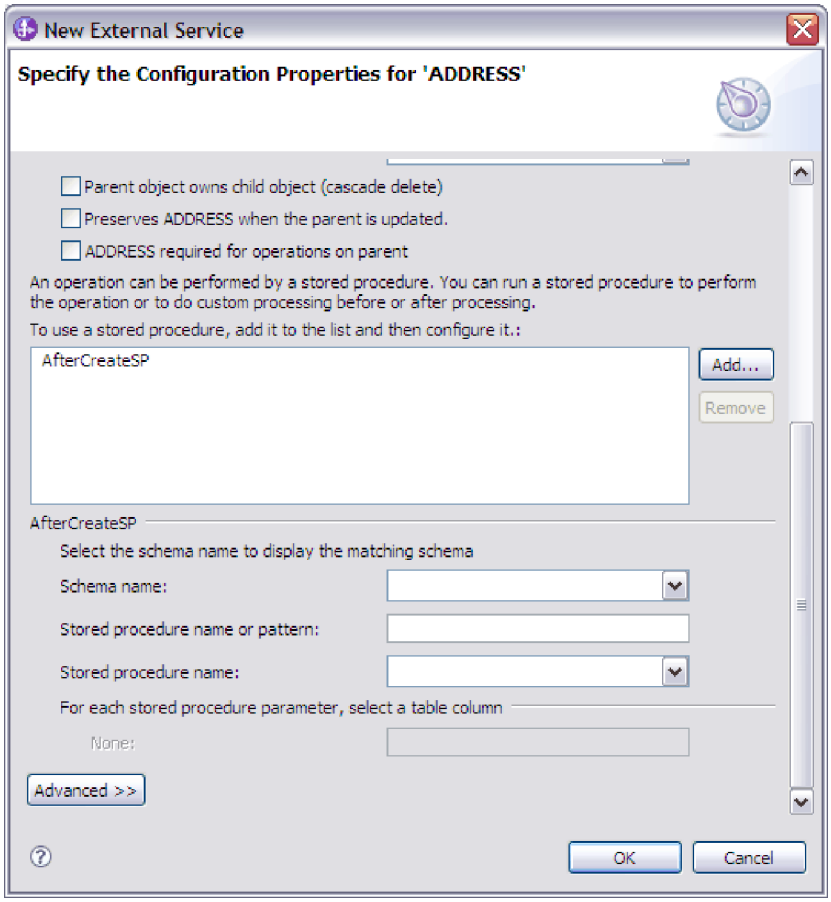
5. An operation can be performed using either a standard SQL statement generated by the adapter or using stored procedures or stored functions from the database. If you want to use stored procedures or stored functions:

   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 these stored procedures: CreateSP, BeforeCreateSP, and AfterCreateSP.

      Note: If you configure the table with RetrieveAllSP, ensure that the stored procedure returns only one result set. Set the ResultSet ASI for the stored procedure to `true` to avoid any of these exceptions being generated at run time: `No resultset found associated with the stored procedure`, `No resultset returned` or `More than one resultset returned`.

   c. Click **OK**. The Specify the Configuration Properties for 'object' 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.

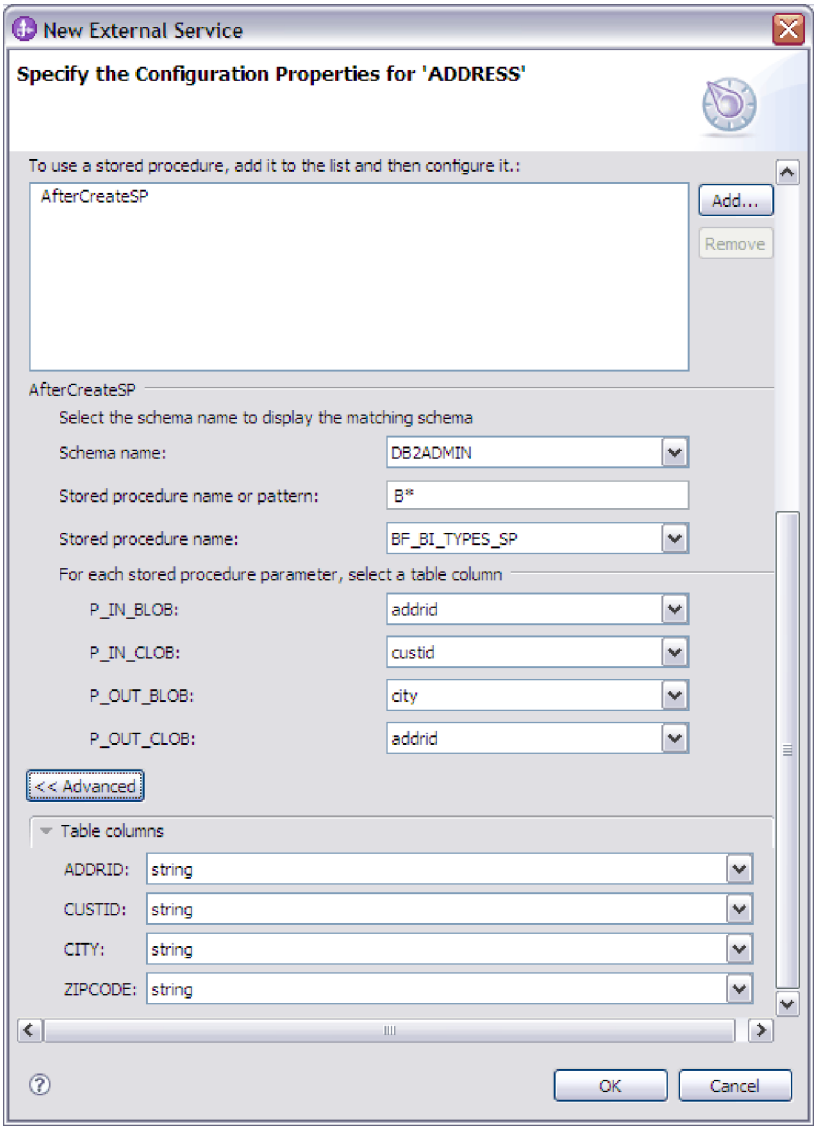


Note: In a hierarchical business object, if you want the stored procedure to be performed for each business object in the hierarchy, you must separately associate a stored procedure with the top-level business object and each child business object or array of business objects. If you associate a stored 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.

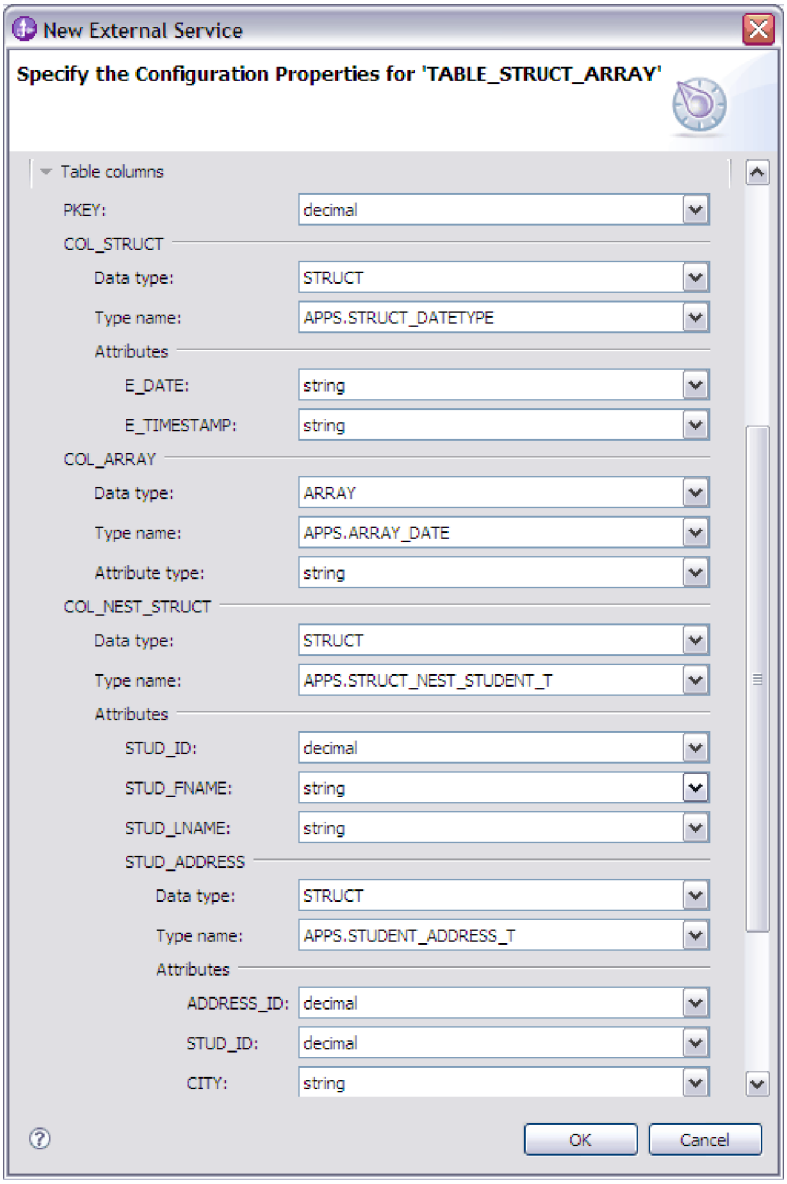
6. For each stored procedure type that you selected, specify the name of the stored procedure in the database and then configure the business object.
   a. In the **Schema name** field, select the name of the schema that contains the stored procedure.
   b. Specify the name of the stored procedure or stored function.
      1) In the **Stored procedure name or pattern** field, either type the name of the stored procedure or stored function, or type a name pattern. Use the question mark or underscore (? or _) to match a single character and the asterisk or percentage sign (* or %) to match multiple characters.
      2) In the **Stored procedure name** field, select the name of the procedure you want.

   The Specify the Configuration Properties for 'object' 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.



7. To specify the data type mapping for each column in the table:
   a. Click **Advanced**.
   b. Expand **Table columns**. For each column in the table, the default data type mapping is displayed. For Oracle databases, if the table contains any complex data type such as an array, structure, nested structure or table, the type name and the sub attribute details are also automatically discovered and displayed. The following figure displays the type name and sub attribute details of an Oracle table containing complex data types.



c. Review the mapping and make changes if required.

Note: If the primary key in a table is of the date or timestamp type, then the object_key in the event_table must be in the 'yyyy-mm-dd hh-mm-ss' format.

8. When all fields in 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 Find Objects in the Enterprise System window.
9. To change the configuration of an object from the **Selected objects** list, select the object name and then click the (Edit) icon.
10. 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

In the Find Objects in the Enterprise System window, continue to select and configure other types of business objects. When you are finished, click **Next** to set

global properties and configure wrapper business objects.

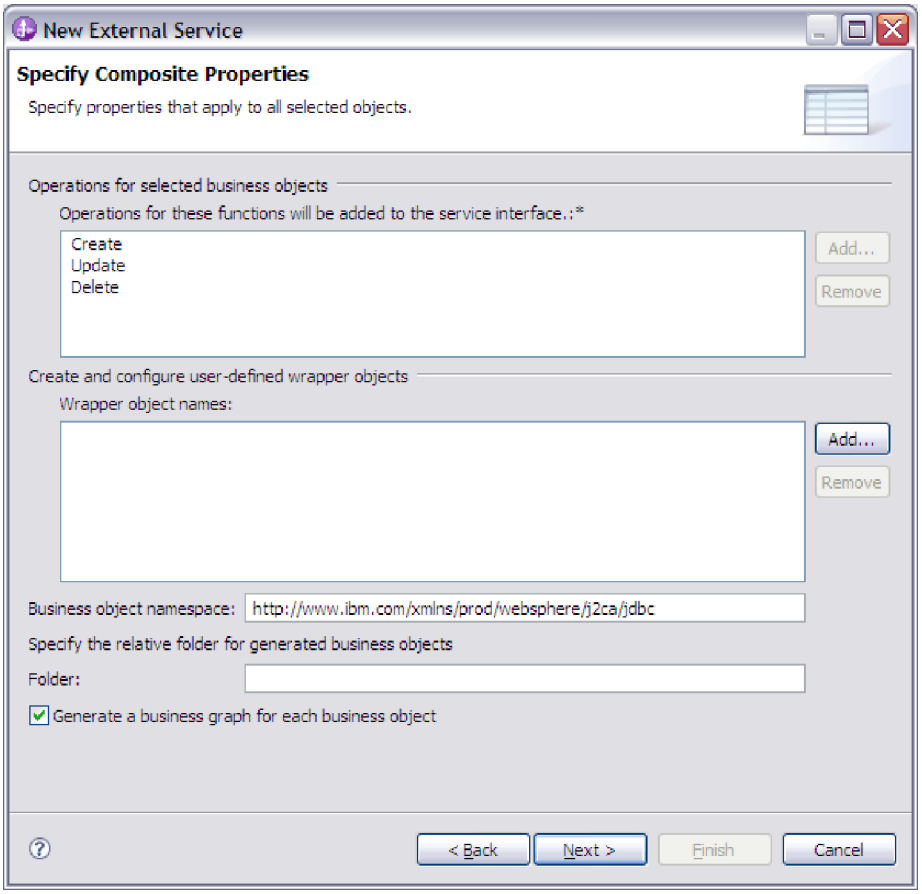
Setting global properties for operations and creating wrapper business objects

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 Find Objects in the Enterprise System window contains all the business objects you want to use in your application, except for wrapper business objects, click **Next**.
2. In the Specify 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.



3. To remove an operation that you do not need, select the operation name and click **Remove**. If reinstate a deleted operation, click **Add** and restore a removed operation.
4. To create a wrapper business object:

   Note: You can use wrapper objects when a table has multiple primary keys and you want to retrieve multiple records from this table. To unset a primary key column use the unsetValueKeyWord property.

   a. In the **Wrapper object names** area, click **Add**.

b. In the Add Value window, type the name of your wrapper business object and then click **OK**. Do not use spaces. The name can contain national language characters.

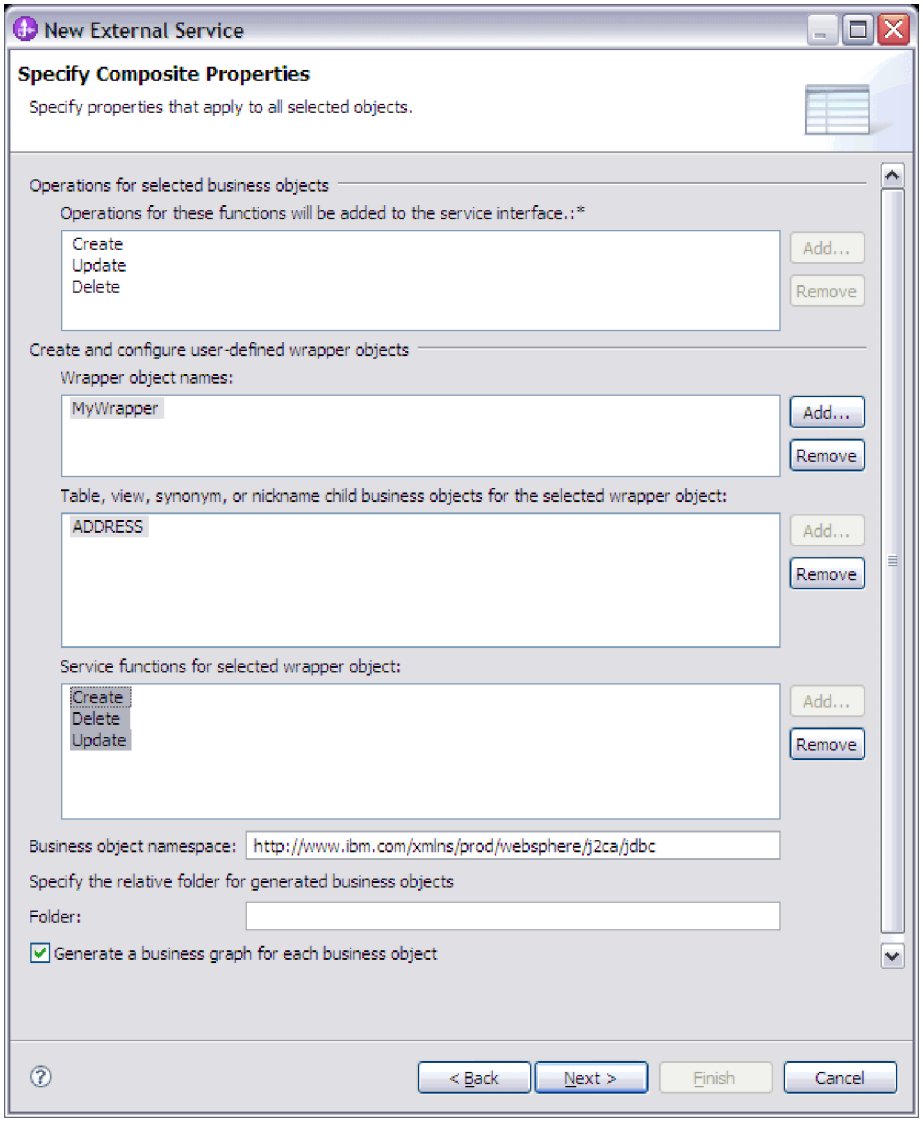
c. In the **Table, view, synonym, or nickname child objects for the selected wrapper object** area, click **Add**.

d. In the Add Value window, select one or more business objects that you want to include in the wrapper and then click **OK**.

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

f. In the Add Value window, select one or more operations that you want to perform on the wrapper object and then click **OK**.

g. Repeat this procedure for each wrapper business object you want to create. The following figure displays the Specify Composite Properties window with one wrapper business objects defined.



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

   The namespace is prefixed to the business object name to keep the business object schemas logically separated. For more information about this property, see "Business object namespace (BONameSpace)" on page 258.

6. Optional: In the **Folder** field, type the relative path to the folder where the generated business objects are to be stored.

   Note: If you are creating multiple adapter artifacts within a module, ensure that you specify different business object folders for each adapter within the module. For example, if you are creating artifacts for Oracle, JDBC, SAP, and JDE within a module, you need to create different relative folders for each of these adapters. If you do not specify different relative folders, the existing artifacts are overwritten when you generate new artifacts.
7. If you want a business graph to be created for each business object, select the **Generate a business graph for each business object** check box. Business graphs are needed only when adding business objects to a module created with a version of WebSphere Integration Developer earlier than version 6.2.x

   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.
8. 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.

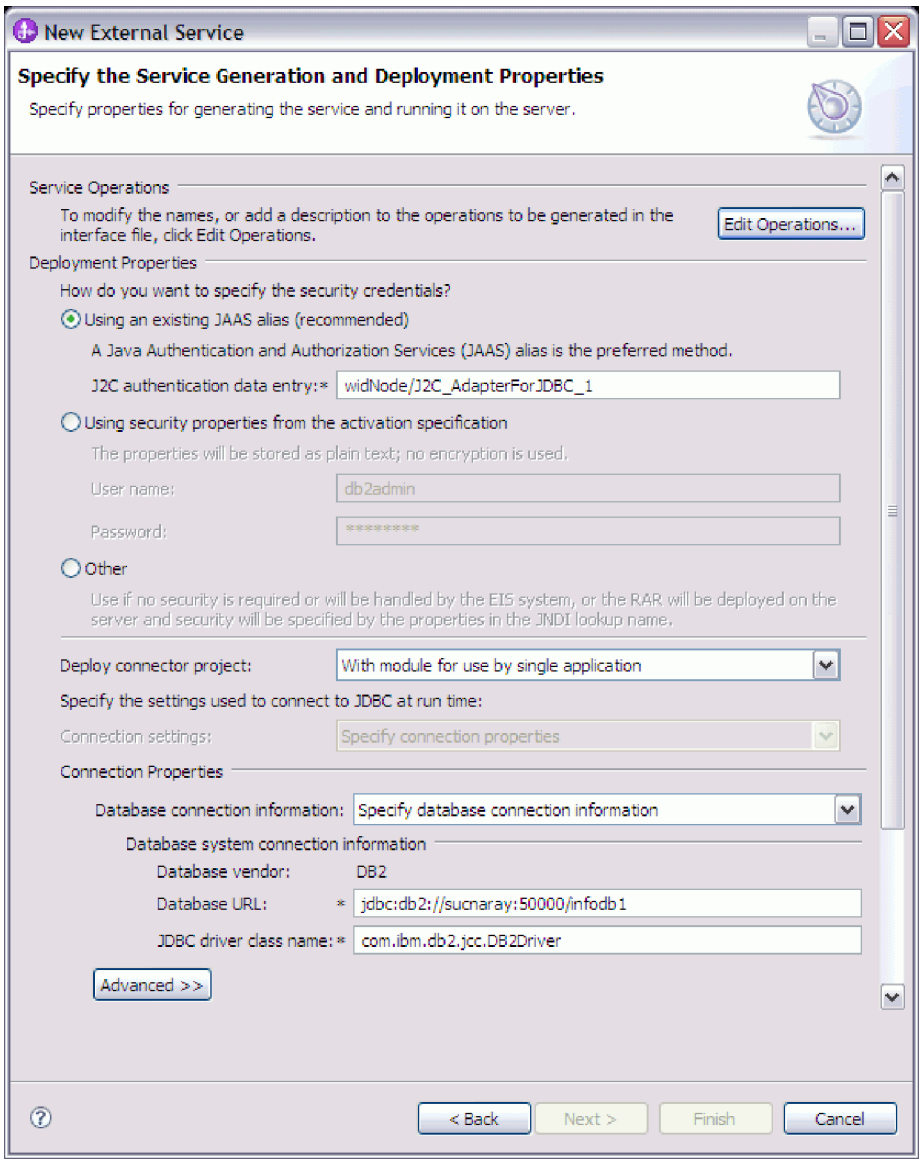
About this task

This task is performed through the Specify the Service Generation and Deployment Properties and Specify the 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 Specify the Service Generation and Deployment Properties window, click **Edit Operations** to review the names of operations or add a description for the operations for the business objects you are creating.



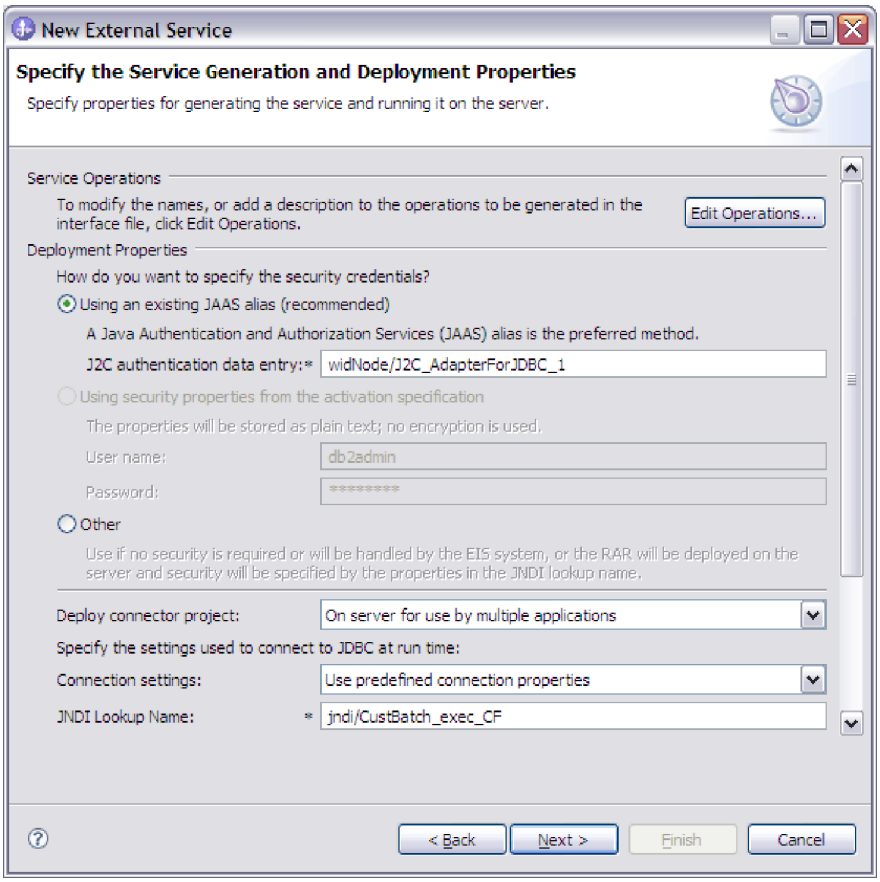
2. In the **Deployment Properties** area, specify how you want the adapter to get the user name and password at run time.
   - To use a J2C authentication alias, click **Using an existing JAAS alias (recommended)** and enter the name of the alias in the **J2C authentication data entry** field. You can 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 managed connection properties, click **Using security properties from the managed connection factory** and enter values in the **User name** and **Password** fields.
   - To administer the user name and password from another mechanism, click **Other**.

   Note: If you use local connection information to establish the database connection, a security credential is required. You can select either **Using an existing JAAS alias (recommended)** or **Using security properties from the managed connection factory** as the security mechanism. If you use an existing data source on the server, the security credential is not required. In this case,

you can select **Other** as the security mechanism. Also, if you set the **J2C authentication data entry** field or set the **User name** and **Password** fields, they override the user name and password in the data source.

3. In the **Deploy connector project** field, specify whether to include the adapter files in the module. Select 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 selected **On server for use by multiple adapters** in the previous step, specify the connection properties to connect at run time.
   - 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. From the **Connection settings** list, select **Use predefined connection properties**.
     b. In the **JNDI Lookup Name** field, 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.



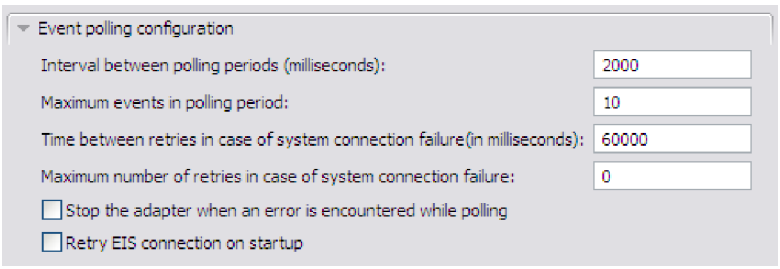
c. Click **Next** to complete this task.

- 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**.

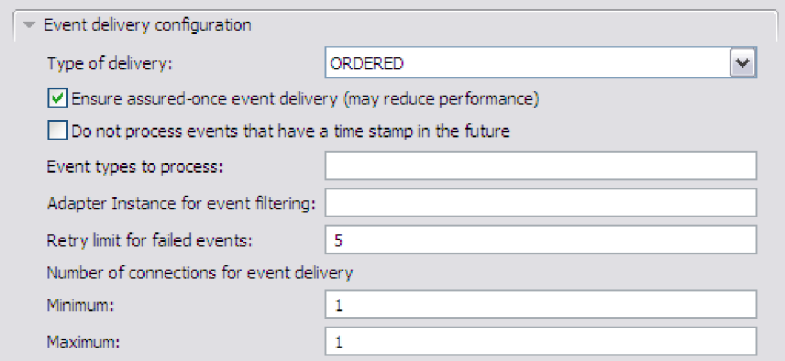
5. In the **Connection Properties** area, specify how you want the adapter to establish database connection at run time.
   - To use an existing data source on the server:
     a. From the **Database connection information** list, select **Specify predefined DataSource**.
     b. In the **Database system connection information** area, enter the name of an existing JNDI data source in the **DataSource JNDI name** field. For more information about this property, see "Data source JNDI name (DataSourceJNDIName)" on page 260.
   - To specify the connection information to be saved in the adapter properties:
     a. From the **Database connection information** list, select **Specify database connection information**.
     b. In the **Database system connection information** area, enter values in the **Database URL** and **JDBC driver class name** fields. For more information about these properties, see "Database URL (DatabaseURL)" on page 261 and "JDBC driver class (JDBCDriverClass)" on page 266.
6. 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 is database-specific. For more information about the properties, see "Activation specification properties" on page 254.

7. Optional: Specify advanced properties by clicking **Advanced**. Expand each of the advanced sections to review the properties.
   - **Event polling configuration**



     a. In the **Interval between polling periods (milliseconds)** field, type the time in milliseconds, that the adapter waits between polling periods. For more information about the property, see "Interval between polling periods (PollPeriod)" on page 269.
     b. In the **Maximum events in polling period** field, type the number of events to deliver in each polling period. For more information about the property, see "Maximum events in polling period (PollQuantity)" on page 270.
     c. In the **Time between retries in case of case of system connection failure (in milliseconds)** field, type the time in milliseconds, that the adapter waits before trying to connect after a connection failure during polling. For more information, see "Retry interval if connection fails (RetryInterval)" on page 270.
     d. In the **Maximum number of retries in case of system connection failure** field, type the number of times the adapter retries the connection before reporting a polling error. For more information, see "Number of times to retry the system connection (RetryLimit)" on page 271.
     e. If you want the adapter to stop if polling errors occur, select the **Stop the adapter when an error is encountered while polling** check box. 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 273.
     f. Select the **Retry EIS connection on startup** check box, if you want the adapter to retry a failed connection when starting. For more information, see "Retry EIS connection on startup (RetryConnectionOnStartup)" on page 271.
   - **Event delivery configuration**



a. In the **Type of delivery** field, select the delivery method. The methods are described in "Delivery type (DeliveryType)" on page 262.

b. If you want to ensure that events are delivered only once and to only one export, select the **Ensure assured-once event delivery** check box. 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 263.

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 the **Do not process events that have a time stamp in the future** check box. For more information, see "Do not process events that have a timestamp in the future (FilterFutureEvents)" on page 263.

d. In the **Event types to process** field, type a comma-separated list of the business objects for which you want the 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`.

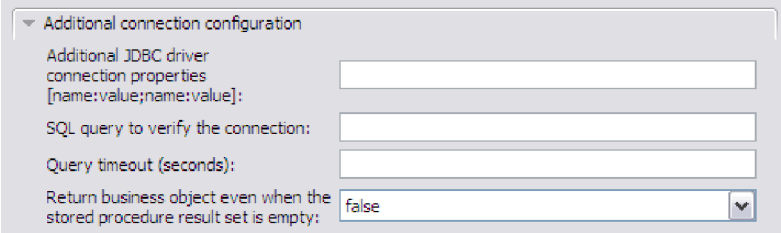
   For more information, see "Event types to process (EventTypeFilter)" on page 265.

e. In the **Adapter Instance for event filtering** field, type the connector id for which you want the events delivered. For more information, see "Adapter Instance for event filtering (AdapterInstanceEventFilter)" on page 256.

f. In the **Retry limit for failed events** field, specify the number of times to try to deliver an event after a delivery failure. For more information, see "Retry limit for failed events (FailedEventRetryLimit)" on page 266.

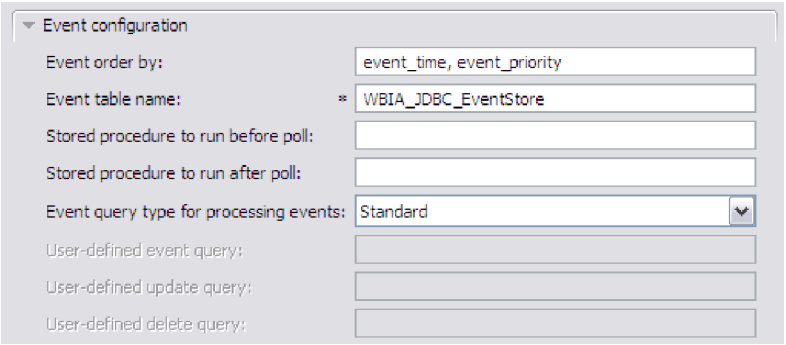
g. In the **Number of connections for event delivery** area, specify the minimum and maximum number of connections to use to deliver events. For more information, see "Minimum connections (MinimumConnections)" on page 268 and "Maximum connections (MaximumConnections)" on page 267.

- **Additional connection configuration**



a. Set **Additional JDBC driver connection properties**. See more information about the property in "Additional JDBC driver connection properties [name:value;name:value] (JDBCDriverConnectionProperties)" on page 258.
b. Set **SQL query to verify the connection**. See more information about the property in "SQL query to verify the connection (PingQuery)" on page 269.
c. In the **Query timeout (seconds)** field, type the length of time, in seconds, that the adapter must wait for a response to a database query. See more information about the property in "Query timeout (seconds) (QueryTimeOut)" on page 270.
d. 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 272.

- **Event configuration**

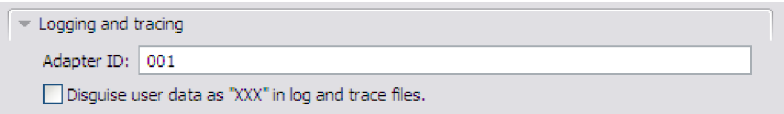


a. In the **Event order by** field, indicate the order in which events are retrieved and processed. This is a comma-separated list of column names from the event table, plus the keywords that control the sort order for each column. Use `asc` for ascending order and `desc` for descending order. For more information, see "Event order by (EventOrderBy)" on page 264.
b. In the **Event table name** field, 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 265.
c. In the **Stored procedure to run before poll** field, specify 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 273.
d. In the **Stored procedure to run after poll** field, 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 273.

e. In the **Event query type for processing events** field, 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
User-defined 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.	"User-defined delete query (CustomDeleteQuery)" on page 258
User-defined event query	The name of the query, stored procedure, or stored function that performs the polling for events.	"User-defined event query (CustomEventQuery)" on page 259
User-defined 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.	"User-defined update query (CustomUpdateQuery)" on page 260

- **Logging and tracing**



a. If you have multiple instances of the adapter, set **Adapter ID** to a value that is unique for this instance. See more information about the property in "Adapter ID (AdapterID)" on page 221.
b. If you want to mask certain information so that the information is not displayed in the logs or traces, select the **Disguise user data as "XXX" in log and trace files** field.

8. Click **Next**. The Specify the Location Properties window is displayed.
9. In the Specify the Location Properties window, specify the name of the module you want to create. This can be a new or existing module.
   - If the 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 is replaced by the new version.
   - Otherwise, create a new module:
     a. Click **New**.
     b. In the Select a Business Integration Project Type window, select the **Module** and click **Next**.
     c. In the Create a 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 the **Use default**

location check box selected. If you want to specify a different folder in the module, clear the option and click **Browse** to specify user a different folder in the **Location** field.

e. If you want the module to be automatically opened in the assembly diagram in WebSphere Integration Developer when the wizard closes, select the **Open the module assembly diagram** check box. Otherwise, clear this option.

f. Click **Finish** to create the new module.

10. 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, select the **Use the default namespace** check box.
    - To specify a different namespace, clear the **Use the default namespace** check box and type a different value in the **Namespace** field.
11. Optional: Specify the folder within the new module where the service description is 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.
12. In the **Name** field, accept the default name of the import or type a different name.
13. Optional: 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**.
14. Optional: In the **Description** field, type a descriptive comment about the module.
15. When you are finished setting properties, click **Finish**.

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

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

- To set the CopyAttribute parameter for a column so that its value is set to the same value as another column.
- To 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 add attributes to a business object. For example, when you run the discovery process against a table in either a DB2 or Microsoft SQL Server database, and that table defines a column as an identity column, the generated business object

for that table does not include the Unique Identifier attribute. Because the adapter needs the Unique Identifier of the identity column at run time, you must add it to the attribute application-specific information. In this case, you add <UID>AUTO</UID> to the attribute application-specific information. For the oracle database, specify the sequence name to UID in order to define a field as an auto-generated field since Oracle does not support identity column.

Note: You do not need to make similar changes to a table business object if you generated it from an Informix database. When you run the discovery process against a table in an Informix database, and that table defines a column as serial (identity columns are referred to as *serial columns* in Informix), the resulting business object does include the Unique Identifier attribute for the serial column, and therefore you need not edit the business object application-specific information. The generated value of the Unique Identifier parameter for a serial column in an Informix database table is either `serial` or `serial8`.

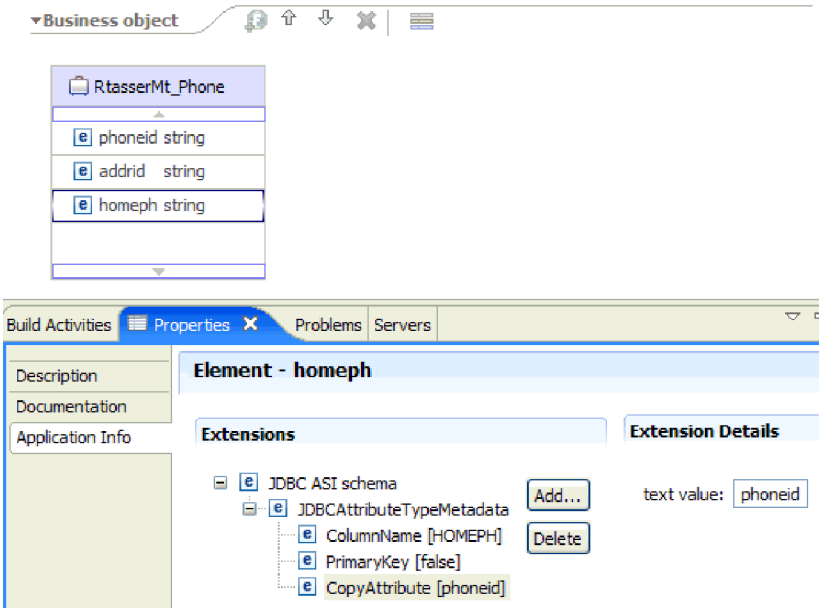
- To configure additional parents for a table business object that has multiple parents. The wizard configures only one parent for a table business object.

This topic provides detailed instructions for setting the CopyAttribute parameter to a table business object. Other changes to business object structures, such as the changes mentioned above, 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 `e-mail` column, set the CopyAttribute parameter of the `contact` attribute to `e-mail`. 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 **CopyAttribute** property.
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 with the CopyAttribute property set to a column in the current table.



Results

The business object is configured to use the CopyAttribute property to set the business object attribute and properties for a database column based on information in another column.

What to do next

You can now test and deploy the module.

Chapter 5. Modifying artifacts

Often business requirements mandate changes to the backend enterprise information system (EIS) data structures. These changes call for regeneration and reconfiguration of artifacts (import and export) that were previously generated using the external service wizard.

A few business scenarios where the output from one enterprise service discovery flow that can be reused in successive flows, are outlined below:

- When you want to add a new object to the object set.
- When you want to modify configurations on selected objects like changes to operations, operation names, and service level settings like security, transactions, and reliability.
- When you want to remove a discovered object from the object set.
- When you want to rediscover existing objects in the service to synchronize the service if the objects in the back-end system have been updated.

To modify existing artifacts, you can invoke the wizard in one of the following ways. The external service wizard is initialized with previously configured settings.

- In the assembly editor, select the component you want to modify, right-click and select **Edit Binding**.
- In the Business Integration view, select the component you want to modify, right-click and select **Edit Binding**.
- Select the component in the assembly editor and select the Properties view. In the Binding tab, click the **Edit** link.

Note: The Edit Binding option is available for artifacts generated using WebSphere Integration Developer 7.0 only. If you are importing a project interchange from an earlier version of WebSphere Integration Developer, the Edit Binding option is not available. If you have made any manual changes to the configuration, running the wizard again will overwrite these changes.

Modifying service import

Modify an import component by rediscovering and reconfiguring the objects using the Edit Binding option in WebSphere Integration Developer.

About this task

You can invoke the external service wizard to modify the information of a service import interface. The wizard automatically populates the existing information for the selected import interface. You can modify the objects and services, and then regenerate the import component with the modified data.

Procedure

1. Invoke the external service wizard for the selected service import interface using one of the following methods.
   - In the assembly editor, select the component you want to modify, right-click and select **Edit Binding**.
   - In the Business Integration view, select the component you want to modify, right-click and select **Edit Binding**.

© Copyright IBM Corp. 2006, 2009

- Select the interface in the assembly editor and select the Properties view. In the Binding tab, click the **Edit** link.

The Find Objects in the Enterprise System window of the external service wizard is displayed. The external service wizard automatically populates the existing configuration details for the selected import interface.

2. In the Find Objects in the Enterprise System window, make the required changes. For more information about discovering objects, see "Discovering database objects" on page 78.

   Note: To change the connection properties for the external service wizard, click **Back** and change the properties in the Specify the Discovery Properties window. For more information see, "Setting connection properties for the external service wizard" on page 75.

   a. You can select and configure the following objects:
      - To configure business objects for tables, views, and synonyms or nicknames for use in your module, see "Selecting and configuring tables, views, and synonyms or nicknames" on page 82.

        Note: If a table selected during the previous enterprise service discovery is deleted in the database, the adapter generates the `Object not found` exception.
      - To configure business objects corresponding to stored procedures and stored functions in the database, see "Selecting and configuring stored procedures and stored functions" on page 89.

        Note: If the stored procedure definition in the database is changed, you must reconfigure the stored procedure and ensure that validation is successful.
      - To configure batch SQL business objects, see "Selecting and configuring batch SQL business objects" on page 92.
      - To configure query business objects, see "Selecting and configuring query business objects" on page 97.

        Note: If the query definition is changed, you must ensure that validation is successful.

   b. To modify the configuration of an object from the **Selected objects** list, select the object name and then click the (Edit) icon.

   c. To remove an object from the **Selected objects** list, select the object name and then click the (Remove) button.

3. Click **Next**. If you click **Cancel**, the changes you made in the previous step does not take effect.
4. In the Specify Composite Properties window, specify properties that apply to all business objects. For more information, see "Setting global properties for operations and creating wrapper business objects" on page 100.
5. Click **Next**.
6. In the Service Generation window, modify the service operations if required.
7. Click **Finish**. The artifacts are regenerated.
8. Complete any other required manual configuration. For more information, see "Completing the configuration" on page 110.

Results

The artifacts are regenerated.

What to do next

You can test and deploy your module.

Modifying service export

Modify an export component by rediscovering and reconfiguring the objects using the Edit Binding option in WebSphere Integration Developer.

About this task

You can invoke the external service wizard to modify the information of a service export interface. The wizard automatically populates the existing information for the selected export interface. You can modify the objects and services, and then regenerate the export component with the modified data.

Procedure

1. Invoke the external service wizard for the selected service export interface using one of the following methods.
   - In the assembly editor, select the component you want to modify, right-click and select **Edit Binding**.
   - In the Business Integration view, select the component you want to modify, right-click and select **Edit Binding**.
   - Select the interface in the assembly editor and select the Properties view. In the Binding tab, click the **Edit** link.

   The Find Objects in the Enterprise System window of the external service wizard is displayed. The external service wizard automatically populates the existing configuration details for the selected export interface.
2. In the Find Objects in the Enterprise System window, make the required changes. For more information about discovering objects, see "Discovering database objects" on page 112.

   Note: To change the connection properties for the external service wizard, click **Back** and change the properties in the Specify the Discovery Properties window. For more information see, "Setting connection properties for the external service wizard" on page 75.

   a. You can select and configure the following objects:
      - To configure business objects for tables, views, and synonyms or nicknames for use in your module, see "Selecting and configuring tables, views, and synonyms or nicknames" on page 116.

        Note: If a table selected during the previous enterprise service discovery is deleted in the database, the adapter generates the `Object not found` exception.

   b. To modify the configuration of an object from the **Selected objects** list, select the object name and then click the (Edit) icon.

   c. To remove an object from the **Selected objects** list, select the object name and then click the (Remove) button.

3. Click **Next**. If you click **Cancel**, the change you made in the previous step does not take effect.
4. In the Specify Composite Properties window, specify properties that apply to all business objects. For more information, see "Setting global properties for operations and creating wrapper business objects" on page 124.
5. Click **Next**.
6. In the Service Generation window, modify the service operations if required.
7. Click **Finish**. The artifacts are regenerated.
8. Complete any other required manual configuration. For more information, see "Completing the configuration" on page 110.

Results

The artifacts are regenerated.

What to do next

You can test and deploy your module.

Chapter 6. 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 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 do not 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.

© Copyright IBM Corp. 2006, 2009

Chapter 7. 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.

Adding external dependencies

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

About this task

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

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.

© Copyright IBM Corp. 2006, 2009

About this task

An adapter needs the JDBC driver files or native system libraries from the database server to be able to communicate with it.

Procedure

1. Obtain the JDBC driver-specific files or 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 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
IBM DB2 for i5/OS	IBM Toolbox for Java remote driver (Type 4)	jt400.jar db2jcc_license_cisuz.jar	None
	IBM Toolkit for Java native driver* (Type 2)	db2_classes.jar	None
IBM DB2 Universal Database for Linux, UNIX, and Windows	IBM DB2 Universal (Type 2)	db2java.zip	None
Oracle	Thin driver	ojdbc6.jar	None
Microsoft SQL Server 2005	Microsoft SQL Server 2005 for JDBC	sqljdbc.jar	None
Informix	IBM Informix JDBC Driver	ifxjdbc.jar ifxjdbcx.jar plus several support .jar files For more information see, IBM Informix JDBC Driver Programmer's Guide	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 Specify the Service Generation and Deployment Properties window.

2. Copy the files to the server.
   - In a testing environment in WebSphere Integration Developer, copy the files to the ${WAS_INSTALL_ROOT}/runtimes/bi_v7/lib/ext directory.
   - In a production environment, copy the files to the ${WAS_INSTALL_ROOT}/lib/ext directory of WebSphere Process Server or WebSphere Enterprise Service Bus.

Adding external software dependencies when the adapter is bundled

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

About this task

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

Procedure

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

Results

The third-party libraries are now a part of your run time environment

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.

   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.
   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**.
   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. Open WebSphere Integration Developer it from the Business Integration perspective by right-clicking the name of your project and then clicking **Test** → **Test Module**.

Procedure

- **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, generate and wire a *target component* to which the adapter sends the events.

About this task

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

Procedure

1. *Conditional:* If there are no servers in the **Servers** view, add and define a new server by performing the following steps:
   a. Place your cursor in the **Servers** view, right-click and select **New → Server**.
   b. From the Define a New Server window, select the server type.
   c. Configure 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 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 about 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 or WebSphere Enterprise Service Bus 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

An adapter needs the JDBC driver files or native system libraries from the database server to be able to communicate with it.

Procedure

1. Obtain the JDBC driver-specific files or 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 files needed for common database software.

Table 13. 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 13. 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 (Type 4)	jt400.jar db2jcc_license_cisuz.jar	None
	IBM Toolkit for Java native driver* (Type 2)	db2_classes.jar	None
IBM DB2 Universal Database for Linux, UNIX, and Windows	IBM DB2 Universal (Type 2)	db2java.zip	None
Oracle	Thin driver	ojdbc6.jar	None
Microsoft SQL Server 2005	Microsoft SQL Server 2005 for JDBC	sqljdbc.jar	None
Informix	IBM Informix JDBC Driver	ifxjdbc.jar ifxjdbcx.jar plus several support .jar files For more information see, IBM Informix JDBC Driver Programmer's Guide	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 Specify the Service Generation and Deployment Properties window.			

2. Copy the files to the server.
   - In a testing environment in WebSphere Integration Developer, copy the files to the ${WAS_INSTALL_ROOT}/runtimes/bi_v7/lib/ext directory.
   - In a production environment, copy the files to the ${WAS_INSTALL_ROOT}/lib/ext directory of WebSphere Process Server or WebSphere Enterprise Service Bus.

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

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

Before you begin

You must set **Deploy connector project** to **On server for use by multiple adapters** in the Specify the Service Generation and Deployment Properties 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. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration → Run administrative console**.
3. Log on to the administrative console.
4. Click **Resources → Resource Adapters → Resource adapters**.
5. In the Resource adapters page, click **Install RAR**.

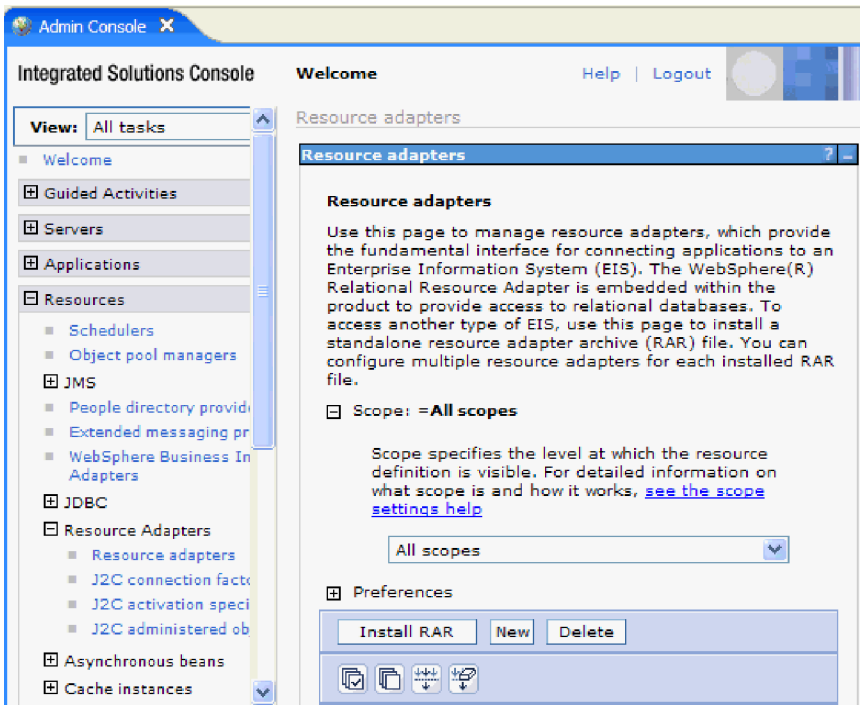
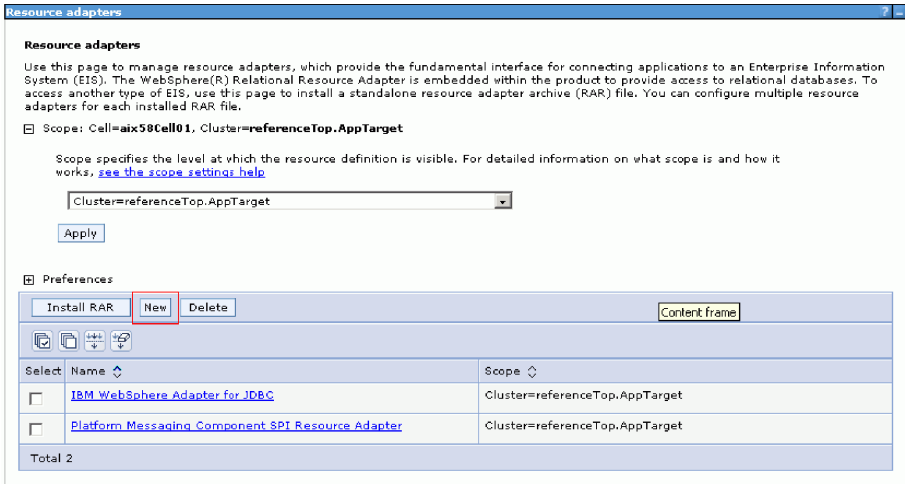


Figure 22. The Install RAR button on the Resource adapters page

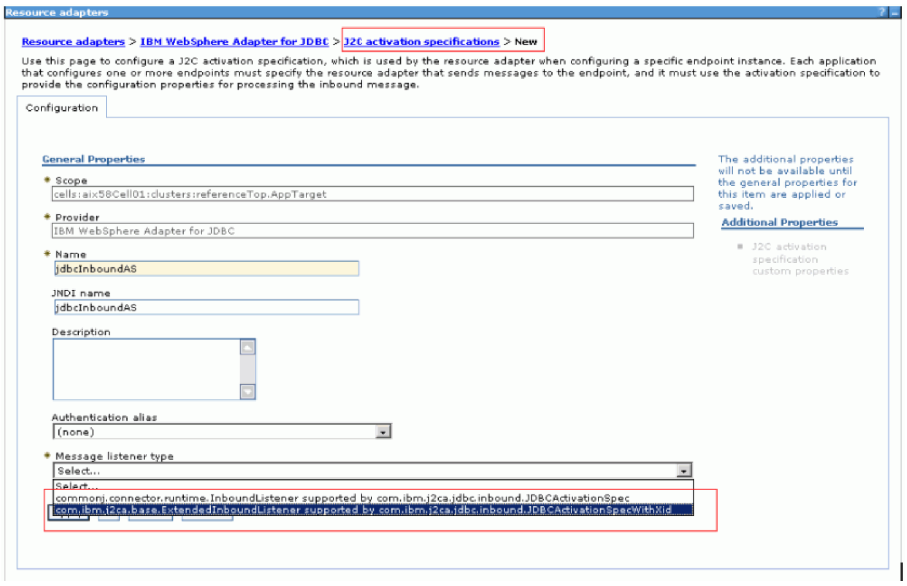
6. In the Install RAR file page, click **Browse** and navigate to the RAR file for your adapter.

   The RAR files are typically installed in the following path:
   WID_installation_directory/ResourceAdapters/*adapter_name*/deploy/*adapter*.rar
7. Click **Next**.
8. Optional: In the Resource adapters page, change the name of the adapter and add a description.
9. Click **OK**.
10. Click **Save** in the **Messages** box at the top of the page.
11. Optional: In the clustered environment, you have to install the adapter in each node separately. After you install the adapter in the clustered environment, complete the following steps:

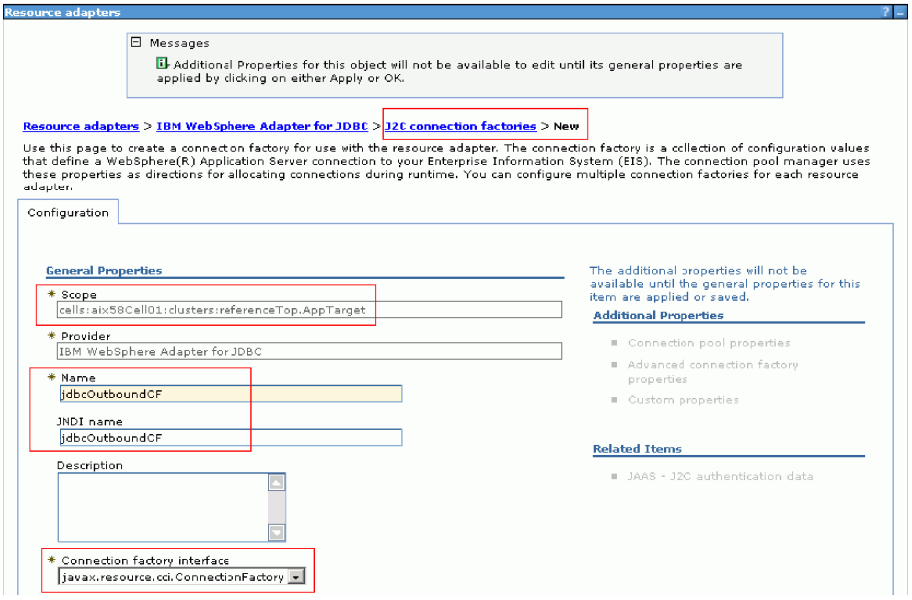
a. Create a JDBC adapter in the clustered environment. Click **Resource adapter** → *target_cluster* → **New**, and create a new resource adapter for a cluster. For example, in the figure below, the target cluster is `Cluster=referenceTop.AppTarget`.



b. Click **Resource adapters** → *adapter_name* → **J2C activation specifications** → **New**, and create a new activation specification for inbound service.



c. Click **Resource adapters** → *adapter_name* → **J2C connection factories** → **New**, and create a new connection factory for outbound service.



d. Configure custom properties, both activation specification and connection factory, on the resource adapter, according to your requirement.

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 **Java EE**.
3. Select **EAR file** and click **Next**.
4. Optional: Select the correct EAR application. The EAR application is named after your module, but with “App” added to the end of the name.
5. Browse for the folder on the local file system where the EAR file will be placed.

6. To export the source files, select the **Export source files** check box. This option is provided in case you want to export the source files in addition to the EAR file. Source files include files associated with Java components, data maps, and so on.
7. To overwrite an existing file, click **Overwrite existing file**.
8. Click **Finish**.

Results

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

What to do next

Install the module in the administrative console. This deploys the module to WebSphere Process Server or WebSphere Enterprise Service Bus.

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 or WebSphere Enterprise Service Bus.

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. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration → Run administrative console**.
3. Log on to the administrative console.
4. Click **Applications → New Application → New Enterprise Application**.

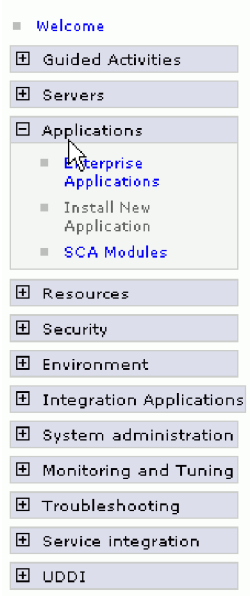


Figure 23. Preparing for the application installation window

5. Click **Browse** to locate your EAR file and click **Next**. The EAR file name is the name of the module followed by "App."
6. Optional: If you are deploying to a clustered environment, complete the following steps.
   a. On the **Step 2: Map modules to servers** window, select the module and click **Next**.
   b. Select the name of the server cluster.
   c. Click **Apply**.
7. Click **Next**. In the Summary page, verify the settings and click **Finish**.
8. Optional: If you are using an authentication alias, complete the following steps:
   a. Expand **Security** and select **Business Integration Security**.
   b. Select the authentication alias that you want to configure. You must have administrator or operator 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 8. 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. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration → Run administrative console**.
3. Log on to the administrative console.
4. Select **Applications → Application Types → WebSphere enterprise application**.
5. From the Enterprise Applications list, click the name of the adapter module whose properties you want to change. The **Configuration** page is displayed.

© Copyright IBM Corp. 2006, 2009

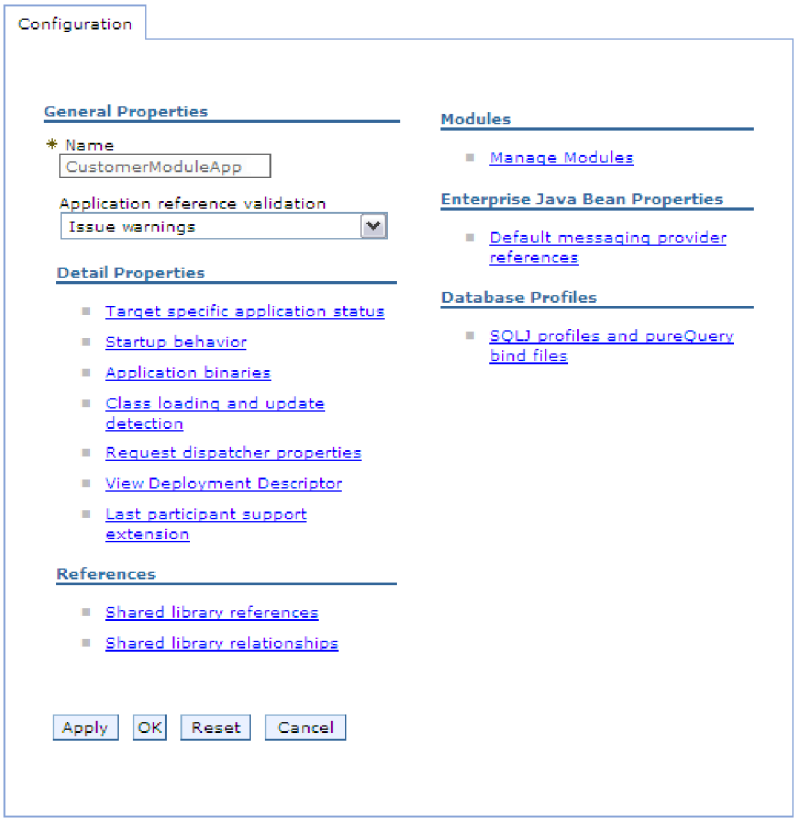


Figure 24. The Manage Modules selection in the Configuration tab

6. Under **Modules**, click **Manage Modules**.
7. Click **IBM WebSphere Adapter for JDBC**.
8. From the **Additional Properties** list, click **Resource Adapter**.
9. On the next page, from the **Additional Properties** list, click **Custom properties**.
10. For each property you want to change, perform the following steps.

    Note: See "Resource adapter properties" on page 220 for more information about these properties.

    a. Click the name of the property. The **Configuration** page for the selected property is displayed.
    b. Change the contents of the **Value** field or type a value, if the field is empty.
    c. Click **OK**.
11. In the Messages area, click **Save**.

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. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration → Run administrative console**.
3. Log on to the administrative console.
4. Select **Applications → Application Types → WebSphere enterprise application**.
5. In the Enterprise Applications list, click the name of the adapter module whose properties you want to change.

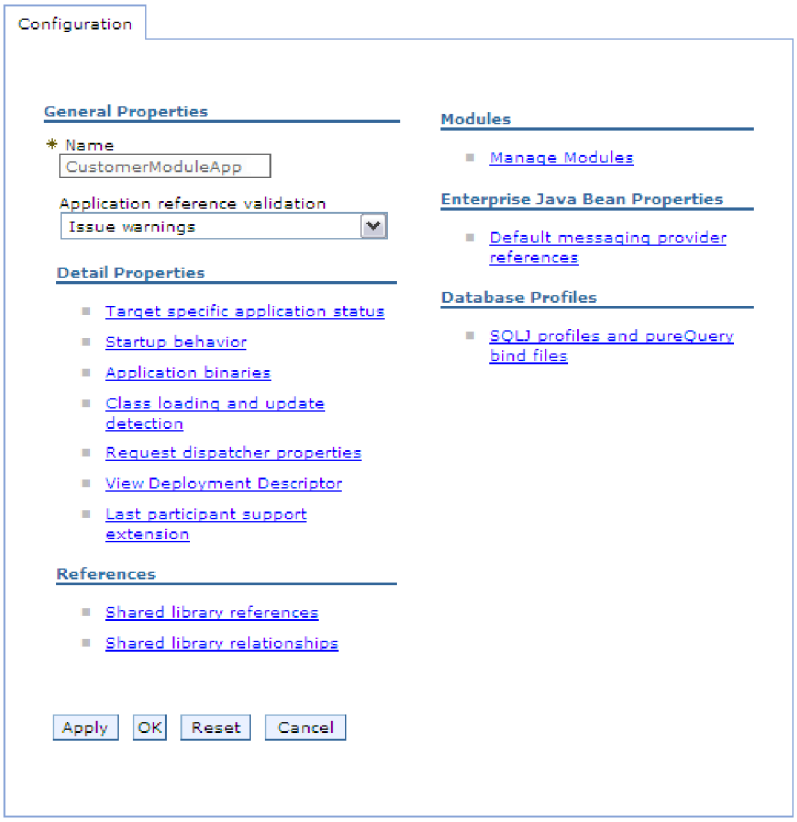


Figure 25. The Manage Modules selection in the Configuration tab

6. Under **Modules**, click **Manage Modules**.
7. Click **IBM WebSphere Adapter for JDBC**.
8. In the **Additional Properties** list, click **Resource Adapter**.
9. On the next page, from the **Additional Properties** list, click **J2C connection factories**.
10. Click the name of the connection factory associated with your adapter module.
11. In 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.
12. For each property you want to change, perform the following steps.

    Note: See "Managed connection factory properties" on page 225 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**.
13. In the Messages area, click **Save**.

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. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration → Run administrative console**.
3. Log on to the administrative console.
4. Select **Applications → Application Types → WebSphere enterprise application**.
5. From the Enterprise Applications list, click the name of the adapter module whose properties you want to change.

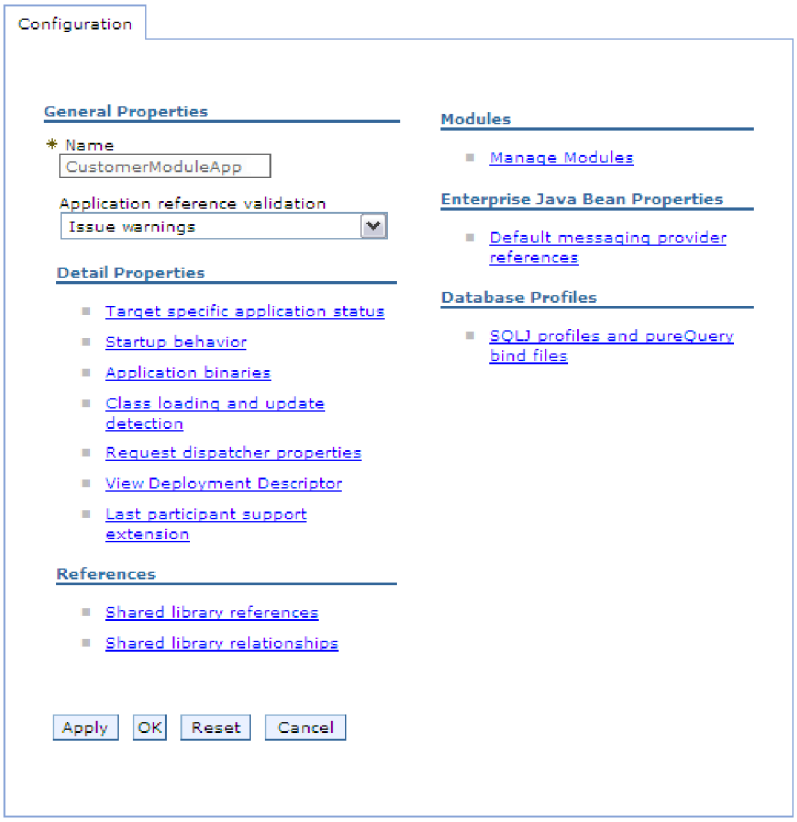


Figure 26. The Manage Modules selection in the Configuration tab

6. Under **Modules**, click **Manage Modules**.
7. Click **IBM WebSphere Adapter for JDBC**.
8. From the **Additional Properties** list, click **Resource Adapter**.
9. On the next page, from the **Additional Properties** list, click **J2C activation specifications**.
10. Click the name of the activation specification associated with the adapter module.
11. From the **Additional Properties** list, click **J2C activation specification custom properties**.
12. For each property you want to change, perform the following steps.

    Note: See "Activation specification properties" on page 254 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**.
13. In the Messages area, click **Save**.

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 is used for outbound operations, you create a connection factory and then set properties for it. If the adapter is 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. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration → Run administrative console**.
3. Log on to the administrative console.
4. Click **Resources → Resource Adapters → Resource adapters**.
5. In the Resource adapters page, click **IBM WebSphere Adapter for JDBC**.
6. In the **Additional Properties** list, click **Custom properties**.
7. For each property you want to change, perform the following steps.

   Note: See “Resource adapter properties” on page 220 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**.
8. In the Messages area, click **Save**.

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. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration → Run administrative console**.
3. Log on to the administrative console.
4. Click **Resources → Resource Adapters → Resource adapters**.
5. In the Resource adapters page, click **IBM WebSphere Adapter for JDBC**.
6. In the **Additional Properties** list, click **J2C connection factories**.
7. If you are going to use an existing connection factory, skip ahead to select from the list of existing connection factories.

   Note: If you have selected **Specify 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 can type `AdapterCF`.
   c. Type a value for **JNDI name**. For example, you can type `com/eis/AdapterCF`.
   d. Optional: Select an authentication alias from the **Component-managed authentication alias** list.
   e. Click **OK**.
   f. In the Messages area, click **Save**.

      The newly created connection factory is displayed.

Figure 27. User defined connection factories for use with the resource adapter

8. In the list of connection factories, click the one you want to use.
9. In 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 225 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. After you have finished setting properties, click **Apply**.
12. In the Messages area, click **Save**.

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. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration → Run administrative console**.
3. Log on to the administrative console.
4. Click **Resources → Resource Adapters → Resource adapters**.
5. In the Resource adapters page, click **IBM WebSphere Adapter for JDBC**.
6. In the **Additional Properties** list, click **J2C activation specifications**.
7. If you are going to use an existing activation specification, skip ahead to select from an existing list of activation specifications.

   Note: If you have 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 can type `AdapterAS`.
   c. Type a value for **JNDI name**. For example, you can type `com/eis/AdapterAS`.
   d. Optional: 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.
8. In the list of activation specifications, click the one you want to use.
9. In the Additional Properties list, click **J2C activation specification custom properties**.
10. For each property you want to set, perform the following steps.

    Note: See "Activation specification properties" on page 254 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. After you have finished setting properties, click **Apply**.
12. In the Messages area, click **Save**.

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. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration → Run administrative console**.
3. Log on to the administrative console.
4. Click **Applications → Application Types → WebSphere enterprise applications**.

   Note: The administrative console is labeled "Integrated Solutions Console".
5. Select the application that you want to start. The application name is the name of the EAR file you installed, without the .EAR file extension.
6. 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. If the server is not running, right-click your server in the **Servers** view and select **Start**.
2. When the server status changes to **Started**, right-click the server and select **Administration → Run administrative console**.
3. Log on to the administrative console.
4. Click **Applications → Application Types → WebSphere enterprise applications**.

   Note: The administrative console is labeled "Integrated Solutions Console".
5. Select the application that you want to stop. The application name is the name of the EAR file you installed, without the .EAR file extension.
6. 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 or WebSphere Enterprise Service Bus.

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 169.
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, and then select the **Enable Performance Monitoring (PMI)** check box.
   d. Select **Custom** to selectively enable or disable statistics.

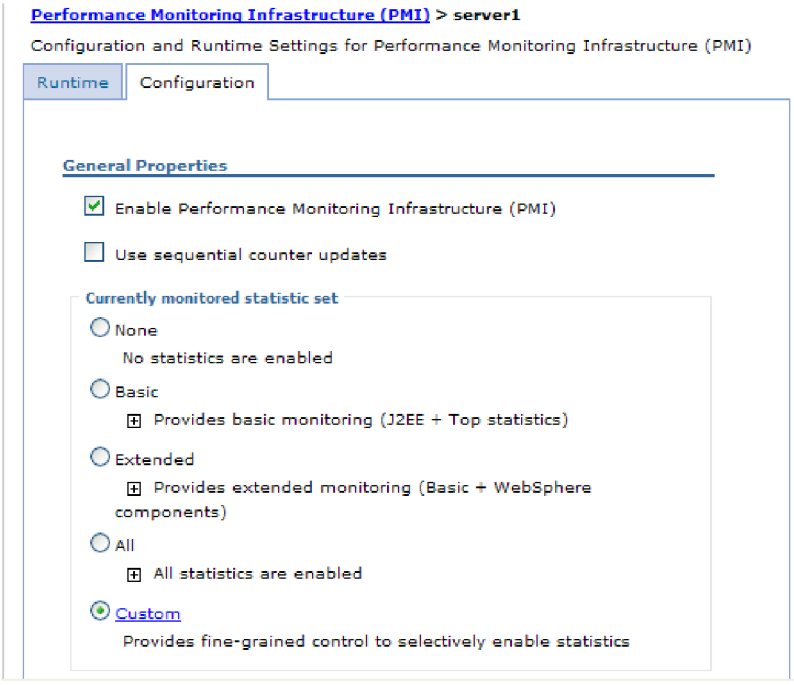


Figure 28. Enabling Performance Monitoring Infrastructure

   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.

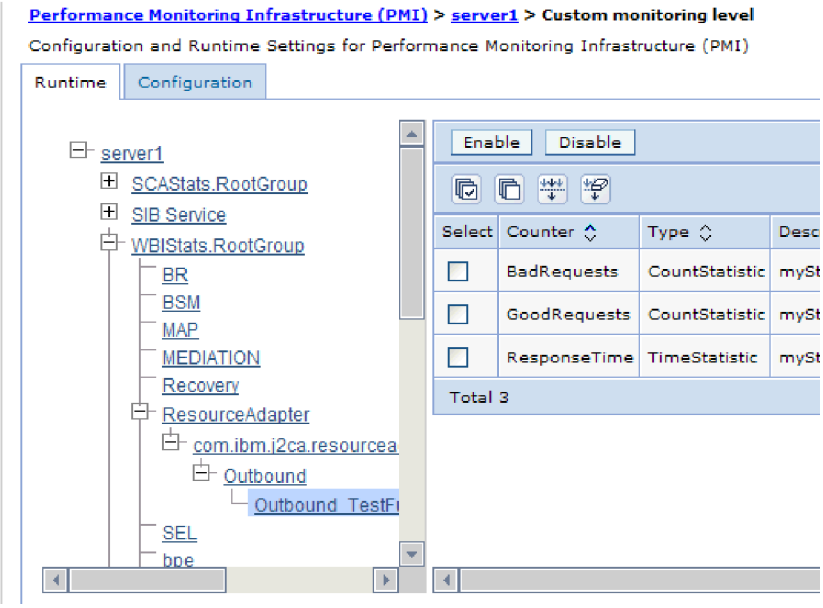


Figure 29. Runtime tab used for configuring PMI

e. Click **WBIStats.RootGroup**. This is a PMI sub module for data collected in the root group. This example uses the name WBIStats for the root group.

f. Click **ResourceAdapter**. This is a sub module 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 or WebSphere Enterprise Service Bus.

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.

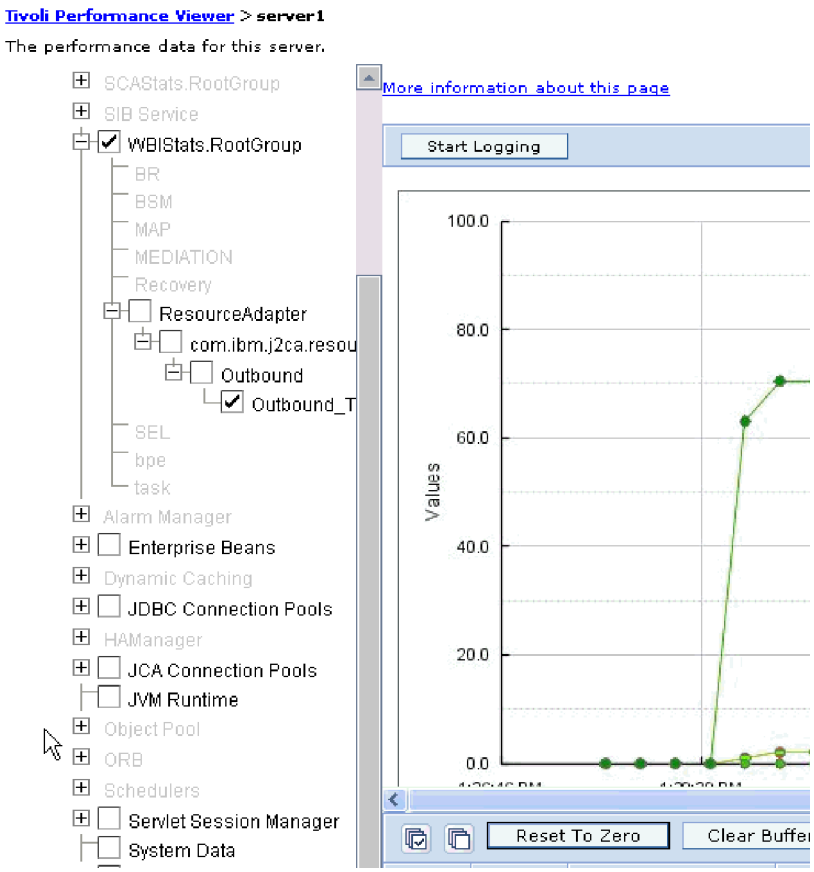


Figure 30. Adapter performance statistics, using graph view

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.

About this task

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, select one of the following options:

     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 the entire 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.

Chapter 9. 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 takes effect when you start or restart the application server. Dynamic, or run time, 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.

© Copyright IBM Corp. 2006, 2009

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.***.
   - For the core component that is common between WebSphere Adapter for JDBC and WebSphere Adapter for Oracle E-Business Suite, select the **com.ibm.j2ca.dbadapter.core.***
   - 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.
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.

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.

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.

Solutions to common problems

Some of the problems you might encounter while running WebSphere Adapter for JDBC with your database are described along with solutions and workaround. 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.

Cannot add object to selection

Problem

At design time, the WebSphere Adapter for JDBC enterprise service discovery process fails when the adapter tries to import a stored procedure from Sybase using the jConnect driver.

Note: The problem does not occur when using the jTDS 1.2.2 driver.

The following message is generated: `Cannot add object to selection: com.sybase.jdbc2.jdbc.SybSQLException: The "CREATE TABLE" is not allowed within a multi-statement transaction in 'tempdb' database.`

Cause

The Auto commit property **Set Auto Commit on database connection** was not selected during enterprise service discovery process and the transaction mode setting for the stored procedure in the Sybase database was set to the default value of "unchained mode". The default "unchained mode" requires explicit begin transaction statements paired with commit transaction or rollback transaction statements to complete the transaction.

Solution

Evaluate the stored procedure definition to determine if you can change it to appropriately process transactions. If you cannot change the stored procedure definition, you can select **Set Auto Commit on database connection** from the Discovery Configuration window of the external service wizard and rerun the discovery process.

By selecting **Set Auto Commit on database connection** you can automatically override the default processing associated with the "unchained mode" configuration. For detailed information about how transaction modes work for the Sybase database, see the Sybase database documentation.

Note: If you select **Set Auto Commit on database connection** from the Discovery Configuration window of the external service wizard, you also need to select **Set Auto Commit on database connection** on the last screen of the external service wizard. The value for **Set Auto Commit on database connection** on the last screen applies to the managed connection factory property used by the adapter at run time to create an outbound connection instance with the database.

CLOB data type of 4K or larger cannot be inserted into Oracle 9i or 10g databases

Problem

The following exception is generated 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 version of the 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

Some business objects that are generated from an Oracle database object do not have 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

Also, column information is not 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.

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 about 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 are 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 reinitialize 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

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 Type 4 and Type 2
- XA data source 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 for Outbound support with a remote DB2 database

Here is the configuration requirements for XA support with the WebSphere Adapter for JDBC with a remote DB2 database.

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, select **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 selected 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, select **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 selected 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, select **Selected → Add Database**.
   c. In the **Instance node** field, select 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, select **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 being processed. The transaction manager will either COMMIT or ROLLBACK these transactions during recovery.

Handling unexpected results from a query SQL statement

If you receive unexpected results from a query, turn the tracing on and look at the query SQL in the log. Turning on tracing is especially helpful when you are in the test client to see if you forgot to *unset* all the unnecessary attributes. It is also practical to turn on tracing to determine if you did fill in your input business object correctly.

Configuring SQL Server 2000 to support XA transaction

To configure SQL SERVER 2000 to support XA transaction:

1. Copy sqljdbc.dll from `Microsoft SQL Server 2000 Driver for JDBC\SQLServer JTA\` to the path*sqlserver_install_directory*`\MSSQL\Binn`.
2. Open the SQL Query Analyzer and run `Microsoft SQL Server 2000 Driver for JDBC\SQLServer JTA\instjdbc.sql`.

WebSphere Adapter for JDBC fails to connect to the SQL Server 2000 using SQL Server 2000 JDBC driver

Cause

This is a limitation with SQL Server 2000 JDBC driver.

Solution

You can use the one of the following approach to resolve this problem:

- Append the value `SelectMethod=Cursor` to the Database URL property. For example: jdbc:microsoft:sqlserver://127.0.0.1:1433;DatabaseName=Partner;SelectMethod=Cursor
- Use the SQL Server 2005 JDBC driver. The SQL Server 2005 JDBC driver provides access to both SQL Server 2000 and SQL Server 2005. For more information, refer to the Microsoft support Web site at http://msdn.microsoft.com/en-us/data/aa937724.aspx.

Multiple JDBC adapter export components in the same SCA module

When multiple JDBC adapter export components exist in the same SCA module, these exports do not pick up and operate on the same event record from same event table. If you define multiple exports to access the same event record, potential problems like database deadlock error might occur. You have to configure different EventTypeFilter (Event types to process) condition on each export for

multiple JDBC Adapter export components picking up records from the same event table so that the risk of one event record being picked up by several exports is eliminated.

Cannot create business object correctly for stored procedures with different specific names in iSeries DB2

Cause

For DB2 iSeries JDBC driver Toolbox for Java™ & JTOpen, the connection property source "metadata source" is used to specify how to retrieve DatabaseMetaData from the database. When it is set to "0", the metadata will be retrieved through the Retrieve Object Information (ROI) data flow. When it is set to "1", the metadata will be retrieved by calling system stored procedures. For version 6.1 and earlier, by default, the metadata source is not set to 1, the database metadata is retrieved through the Retrieve Object Information (ROI) data flow. This causes the problem.

The default for the DB2 iSeries version 7.1 is to retrieve database metadata by calling stored procedures.

Here is the detail information about the database metadata in the `jtopen_6_4_source\com\ibm\as400\access\doc-files\JDBCProperties.html` from the JTOpen download Web site : http://sourceforge.net/projects/jt400/files/JTOpen-full/6.4/

Solution

Add the metadata `source=1` to the connection URL and ensure that there is no space after ";" when connecting to DB2 iSeries version 6.1 and earlier. For example, jdbc:as400://wsbcas12.rtp.raleigh.ibm.com;metadata source=1

Changing the schema without regenerating the artifacts

If only schema is changed and table name is not changed, edit the *.export file for inbound or *.import file for outbound, and *.xsd files and change the original schema to the new schema as described below:

To make changes to the import or export file:

1. In the Java perspective open the *.import or *.export file in the text editor. For example, JDBCInboundInterface.export.

   Below is the code snippet for JDBCInboundInterface.export

```
<connection type="com.ibm.j2ca.jdbc.inbound.JDBCActivationSpecWithXid"
listenerType="com.ibm.j2ca.base.ExtendedInboundListener"
selectorType=
"com.ibm.j2ca.extension.emd.runtime.StructuredDataFunctionSelector">
      <properties>
        <databaseURL>jdbc:oracle:thin:@localhost:1521:ORA92
</databaseURL>
        <databaseVendor>ORACLE</databaseVendor>
        <jdbcDriverClass>oracle.jdbc.driver.OracleDriver
</jdbcDriverClass>
        <password>jcajdbc</password>
        <returnDummyBOForSP>false</returnDummyBOForSP>
        <userName>jcajdbc</userName>
</properties>
    </connection>
```

2. Change the existing user name and password to the user name and password used by new schema.

To make changes to the *.xsd files:

1. Open the *.xsd file in the text editor. For example, JcajdbcCustomer.xsd
   Below is the code snippet for JcajdbcCustomer.xsd:

```
<jdbcasi:TableName>JCAJDBC.CUSTOMER</jdbcasi:TableName>
<jdbcasi:Operation>
<jdbcasi:Name>Retrieve</jdbcasi:Name>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>BeforeRetrieveSP
</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>JCAJDBC.fn_beforeRetrievesSP
</jdbcasi:StoredProcedureName>
<jdbcasi:ReturnValue>RS</jdbcasi:ReturnValue>
<jdbcasi:Parameters>
<jdbcasi:Type>IP</jdbcasi:Type>
<jdbcasi:PropertyName>pkey</jdbcasi:PropertyName>
</jdbcasi:Parameters>
</jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>AfterRetrieveSP
</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>JCAJDBC.sp_afterRetrieveSP
</jdbcasi:StoredProcedureName>
<jdbcasi:Parameters>
<jdbcasi:Type>IP</jdbcasi:Type>
<jdbcasi:PropertyName>fname</jdbcasi:PropertyName>
</jdbcasi:Parameters>
<jdbcasi:Parameters>
<jdbcasi:Type>OP</jdbcasi:Type>
<jdbcasi:PropertyName>RS</jdbcasi:PropertyName>
</jdbcasi:Parameters>
</jdbcasi:StoredProcedures>
</jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
```

2. Update the schema name prefix for table and procedure names (if you defined in BO) with the new schema name. In the example, change JCAJDBC.CUSTOMER to <NewSchema>.CUSTOMER, change JCAJDBC.fn_beforeRetrievesSP to <NewSchema>.fn_beforeRetrievesSP, change JCAJDBC.sp_afterRetrieveSP to <NewSchema>.sp_afterRetrieveSP.

Result set does not contain any values when validating the stored procedure

Problem

When the JDBC adapter is configured to invoke a stored procedure on MS SQL Server 2000, the wizard does not return any values from the result set after selecting the "validate the syntax..." check box.

Solution

Either set AutoCommit ='true' or upgrade your driver to SQL Server version 2005.

Stored procedure name BIDI support for MS SQL Server

Problem

For MS SQL server, if the stored procedure name is in a language written from right-to-left, the business object configuration window for the stored procedure is not displayed. The log file displays the exception "java.lang.RuntimeException: com.microsoft.sqlserver.jdbc.SQLServerException: Conversion failed when

converting the nvarchar value '??' to data type int". If the EMD wizard is completed ignoring this exception, the corresponding stored procedure business object is not generated.

Solution

The stored procedure name must be in a language written from left-to-right. Change the language in which the stored procedure name is represented.

Schema name BIDI support for Oracle database

Problem

For Oracle database, if the schema name is in a language written from right-to-left, the stored procedure validation fails. The log file displays the exception "java.sql.SQLException: ORA-06550: line xx, column xx: PLS-00302: component 'xxx' must be declared". If the EMD wizard is completed ignoring this exception, the corresponding stored procedure business object is generated. However, when the corresponding outbound operation is processed an exception is generated.

Solution

The schema name must be in a language written from left-to-right. Change the language in which the schema name is represented.

WebSphere Adapter for JDBC fails when a table name or column name contains SQL keywords or other special characters

Problem

When the table or column name contains special characters like space, single quotation mark, or SQL reserved keywords, WebSphere Adapter for JDBC fails.

Cause

There are table names with special characters, for example, in Oracle server, "TABLE", "TABLE 2", "table", "table 2", "'TABLE'", "'TABLE 2'", or there are columns in a table with special characters, for example, "DESC", "DESC 2", "desc", "desc 2", "'COLUMN'", "'COLUMN 2'".

Solution

The terms "TABLE" and "DESC" are reserved SQL keywords. As these table names or column names are not handled correctly during the processing of the SQL statement, you have to edit the `TableName` or `ColumnName` application-specific information and enclose table names or columns with double quotation marks.

Here is an example in which the table name and column name contain a SQL keyword and a special character (space):

```
<jdbcasi:TableName>YUANJS.table 2</jdbcasi:TableName>
<jdbcasi:ColumnName>DESC</jdbcasi:ColumnName>
```

Edit the ASI to include the table name and column name with double quotation marks.

```
<jdbcasi:TableName>"YUANJS.table 2"</jdbcasi:TableName>
<jdbcasi:ColumnName>"DESC"</jdbcasi:ColumnName>
```

WebSphere Adapter for JDBC and MySQL database support

During metadata discovery, the JDBC Enterprise Metadata Discovery (EMD) first retrieves the schema information from the database-specific JDBC driver and then depending on schema selection, the information about other database entities such as tables, procedures, views, and synonyms are retrieved. However, in the case of MySQL, the JDBC driver does not return schema information. This is because MySQL JDBC driver implementation does not return the information about database schemas in versions before version 5.0.4. Hence, you cannot proceed further with the discovery and artifact generation.

Sample SQL script for generating tables and triggers on i5/OS

Problem

The JDBC J2EE Connector architecture (JCA) resource adapter ships a sample script for DB2 called scripts_db2.sql. For UDB on i5/OS, the SQL syntax in the DB2 sample script does not function as it is written.

Cause

UDB requires fully qualified schema SQL in the form SCHEMA.TABLE. The current script does not contain qualified schema.

Solution

Modify scripts_db2.sql for use on i5/OS to include the fully qualified schema name with the table and trigger names of the table and trigger scripts. For example, below is a sample table creation script from scripts_db2.sql:

```
CREATE TABLE customer
  (
   pkey VARCHAR(10) NOT NULL PRIMARY KEY,
   fname VARCHAR(20),
   lname VARCHAR(20),
   ccode VARCHAR(10)
);
```

For use on i5/OS, modify the script and replace <schema_name> with the fully qualified schema name as shown below:

```
CREATE TABLE <schema_name>.customer
(
  pkey VARCHAR(10) NOT NULL PRIMARY KEY,
  fname VARCHAR(20),
  lname VARCHAR(20),
  ccode VARCHAR(10)
);
```

WebSphere Adapter for JDBC locks rows in the table causing other applications to timeout

Problem

WebSphere Adapter for JDBC was included as part of a global transaction accessing shared data in database tables. This shared data in the database tables was also accessed by another application. As part of the global transaction, the WebSphere JDBC adapter held the locks on the rows, thus causing the other applications to timeout when trying to access the same shared data.

Solution

Remove WebSphere Adapter for JDBC from the global transaction if you are accessing shared data. Instead, use some mechanism at the business level to ensure the business data is consistent. For example, use a variable to indicate the operation on the shared data is successful or not and log the critical operation on this data.

Incorrect binary data is returned when querying DB2/AS400

Problem

When performing a query against DB2/AS400 using WebSphere Adapter for JDBC version 6.1, the binary fields are returned as EBCDIC characters on i5/OS. For example, the field returns 4040f9f9f1f0f0f0f1f7f6f1 instead of 9910001761. The field in the AS400 is defined as BINCHAR and tagged with CCSID 65535.

Cause

The field in the i5/OS database is tagged with CCSID 65535. The toolbox JDBC driver recognizes this CCSID as a field that should not be translated.

Solution

Tag fields that you want to be translated with a valid CCSID. Alternately, you can set the "translate binary" connection property to "true". This instructs the JDBC driver to translate all fields, including those tagged with CCSID 65535. A quick way to do this is to add ";translate binary=true" to the end of the URL used when connecting to the database.

Error when using stored procedure with return value

Problem

WebSphere Adapter for JDBC fails when processing stored procedures that have a return value. The error message generated would be something like: Procedure 'GetPolicyCount' expects parameter '@count', which was not supplied.

Cause

The adapter currently does not support processing stored procedures with return value. The example below highlights the lines in the XSD generated by the Enterprise Metadata Discovery which causes the error, due to lack of support in the adapter.

When using a stored procedure that returns a status value, in the XSD that is generated, we have the following references to the return value (given in bold) in the example below:

```
<?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/pocusergetpolicycount"
xmlns:pocusergetpolicycount=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/pocusergetpolicycount"
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="PocuserGetpolicycount">
<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:TableName>POCUser.GetPolicyCount</jdbcasi:TableName>
<jdbcasi:StatusColumnName>returnvalue</jdbcasi:StatusColumnName>
<jdbcasi:StatusValue></jdbcasi:StatusValue>
<jdbcasi:Operation>
<jdbcasi:Name>Retrieve</jdbcasi:Name>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>RetrieveSP</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>GetPolicyCount</jdbcasi:StoredProcedureName>
<jdbcasi:Parameters>
<jdbcasi:Type>OP</jdbcasi:Type>
<jdbcasi:PropertyName>returnvalue</jdbcasi:PropertyName>
</jdbcasi:Parameters>
<jdbcasi:Parameters>
<jdbcasi:Type>IP</jdbcasi:Type>
<jdbcasi:PropertyName>ipolicynum</jdbcasi:PropertyName>
</jdbcasi:Parameters>
</jdbcasi:StoredProcedures>
</jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="returnvalue" type="int" minOccurs="1" 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:ColumnName>@RETURN_VALUE</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>false</jdbcasi:PrimaryKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
<element name="ipolicynum" 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:ColumnName>@iPolicyNum</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>false</jdbcasi:PrimaryKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>
```

Solution

Delete all references to the return value (the ones given in bold above) from the XSD file for the stored procedure execution to work without errors. The modified XSD in the example above would now look like the following:

```
<?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/pocusergetpolicycount"
xmlns:pocusergetpolicycount=
"http://www.ibm.com/xmlns/prod/websphere/j2ca/jdbc/pocusergetpolicycount"
```

```
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="PocuserGetpolicycount">
<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:TableName>POCUser.GetPolicyCount</jdbcasi:TableName>
<jdbcasi:StatusColumnName>returnvalue</jdbcasi:StatusColumnName>
<jdbcasi:StatusValue></jdbcasi:StatusValue>
<jdbcasi:Operation>
<jdbcasi:Name>Retrieve</jdbcasi:Name>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>RetrieveSP</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>GetPolicyCount</jdbcasi:StoredProcedureName>
<jdbcasi:Parameters>
<jdbcasi:Type>IP</jdbcasi:Type>
<jdbcasi:PropertyName>ipolicynum</jdbcasi:PropertyName>
</jdbcasi:Parameters>
</jdbcasi:StoredProcedures>
</jdbcasi:Operation>
</jdbcasi:JDBCBusinessObjectTypeMetadata>
</appinfo>
</annotation>
<sequence minOccurs="1" maxOccurs="1">
<element name="ipolicynum" 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:ColumnName>@iPolicyNum</jdbcasi:ColumnName>
<jdbcasi:PrimaryKey>false</jdbcasi:PrimaryKey>
</jdbcasi:JDBCAttributeTypeMetadata>
</appinfo>
</annotation>
</element>
</sequence>
</complexType>
</schema>
```

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 the 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 typically 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. This 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 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.`

Solution

Run the scripts `initxa.sql` and `initjvm.sql` that are included in the Oracle directory.

```
<ORACLE_HOME>javavm\install
file: initxa.sql
file: initjvm.sql
```

This activity is likely 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 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. It might not run successfully if the database memory space is too small.

To fix this, run the initjvm.sql script. This indicates which parameters need to be adjusted. The parameters are stored in the following file:

```
<ORACLE_HOME>\database
file: init<DATABASE_SID>.ora
```

Table 14 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

ResourceException during outbound processing

If you get a ResourceException, examine the root cause field to determine the cause. Common problems have the following root causes:

- SQLException exception

  If the SQLException includes the text `UserID or password is invalid`, then the user ID or password specified for the outbound connection is 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 ConnectException includes the text `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 during inbound processing

This exception indicates that there is a repeated problem connecting to the database. To polls for events, the adapter must connect to the database. If the connection fails, the adapter waits for the configured time before trying to connect to the database again. The adapter tries to connect the configured number of times before it stops polling. When the adapter stops polling, it generates the ResourceException.

RecordNotFoundException is generated when processing a RetrieveAll or Retrieve operation in test client

Problem

When processing a RetrieveAll or Retrieve operation in the WebSphere® Integration Developer test client, if the attributes that are not needed in the WHERE condition (for the SELECT statement) are left blank, the RetrieveAll or Retrieve operation fails, and the following exception may be generated: `RecordNotFoundException: Record not found in EIS`.

Solution

In the test client, set the values of the attributes that are required to `<unset>`. Process the RetrieveAll operation. If the exception is generated again, it is likely that no matching records exist in the database table.

WebSphere Adapter for JDBC throws RecordNotFoundException

Problem

When the adapter tries to retrieve data from the database and there is no entry for the requested keys, the adapter throws the following exception instead of returning an empty or null object: 5/13/09 12:28:29:332 GST com.ibm.j2ca.base.exceptions.RecordNotFoundException.

Cause

When processing the RetrieveAll operation, the adapter generated the RecordNotFoundException when no records are returned from the database. The WebSphere Adapter for JDBC does not handle the empty entry correctly.

Problem

Contact IBM support to retrieve WebSphere Adapter for JDBC interim fix V6.0.2.3IF10. With this fix, a new property called errorOnEmptyResultset has been added on the Managed Connection Factory. The default value for this property is "true" and if no rows are returned for the RetrieveAll operation, a RecordNotFoundException is thrown.

To override the behavior of RetrieveAll operation, you can change the value for the errorOnEmptyResultset property through the admin console and set it to False after deploying the application. The adapters returns an empty container when no records are found after processing the RetrieveAll operation. This property is not configurable through the EMD in this fix, hence it has to be changed in the admin console.

You can also add the "errorOnEmptyResultset" property to *.import file as shown below:

```
<connection type="com.ibm.j2ca.jdbc.JDBCManagedConnectionFactory"
interactionType="com.ibm.j2ca.jdbc.JDBCInteractionSpec">
<properties>
<databaseURL>
jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=adapter
</databaseURL>
<jdbcDriverClass>
com.microsoft.jdbc.sqlserver.SQLServerDriver</jdbcDriverClass>
```

```
<password>adrienne</password>
<userName>sa</userName>
<errorOnEmptyResultset>false</errorOnEmptyResultset>
</properties>
```

JDBC JCA adapter 6.x throws exception when no records are found while retrieving data

See "WebSphere Adapter for JDBC throws RecordNotFoundException" on page 189.

ResultSet ASI for stored procedures requires manual intervention for returning a result set

Problem

WebSphere Adapter for JDBC enterprise metadata discovery process does not set the ResultSet ASI for stored procedures that return a result set and are attached to a business object as a verb ASI.

Cause

Only the Oracle database returns a result set as an output parameter. For all other databases, there is no way to determine if the stored procedure returns a result set.

Solution

For all stored procedures that return a result set and are attached to a business object as a verb ASI, manually set the ResultSet ASI to true to indicate that the stored procedure returns a result set.

The example below shows how the ASI must be set in the business object.

```
<jdbcasi:Operation>
<jdbcasi:Name>RetrieveAll</jdbcasi:Name>
<jdbcasi:StoredProcedures>
<jdbcasi:StoredProcedureType>RetrieveAllSP</jdbcasi:StoredProcedureType>
<jdbcasi:StoredProcedureName>SCOTT.GETCUSTS1</jdbcasi:StoredProcedureName>
<jdbcasi:ResultSet>true</jdbcasi:ResultSet>
<jdbcasi:Parameters>
```

Enterprise service discovery cannot save the values of MaxNumOfRetRS and ResultSet ASI in the generated business object

Solution

In the wizard, if you first set the value for the **The maximum number of ResultsSets returned from the stored procedure** field and validate the stored procedure, the value in the **The maximum number of ResultsSets returned from the stored procedure** field is automatically reset to 0 ignoring the value you had earlier. For the values to be saved correctly in the generated business object, first validate the stored procedure successfully and then specify a value in the **The maximum number of ResultsSets returned from the stored procedure** field.

See also, Selecting and configuring stored procedures and stored functions.

WebSphere Adapter for JDBC does not support multiple stored procedures with the same name in one schema

See Selecting and configuring stored procedures and stored functions.

Date type ASI for business object attributes refers to columns of type Date and Time

Problem

The JDBC Enterprise Metadata Discovery maps all Time, TimeStamp, and Date database columns to attributes of Service Data Objects type Date in the business object. The SDO date types are handled in such a way that the Date type attributes are displayed in GMT instead of the actual time stored in the database. The time portion is lost when values are set on an SDO or on a database.

Solution

For those columns whose SQL type is DATE, TIME, and TIMESTAMP, JDBC Enterprise Metadata Discovery sets the corresponding attribute type to string, and an attribute application-specific information named DataType is added. Its value is set to either Date, Time, or TimeStamp. To generate the business object XSDs with the proper attribute type and application-specific information, run the JDBC Enterprise Metadata Discovery again.

When setting the date and time values on the business object during outbound, use the following formats:

- The date format is yyyy-mm-dd
- The time format is hh:mm:ss
- The timestamp format is yyyy-mm-dd hh:mm:ss

Similarly, for inbound, if the key value is of the date or time type for the object_key column in the event table, the values have to be entered in the format as described above.

XML data types and XQueries are not directly supported by J2CA JDBC adapter

See How does J2CA Adapter work with XML data type in DB2

BPEL cannot catch specific business fault

See BPEL cannot catch specific Business Fault

Outbound fails due to an exception in global transaction

Problem

Outbound fails due to an exception in global transaction and the transaction cannot not be rolled back successfully.

Cause

From version 6.2 onwards, WebSphere Adapter for JDBC supports business objects for exceptions including IntegrityConstraintViolationException, MatchesExceededLimitException, MissingDataException,

MultipleMatchingRecordsException, ObjectNotFoundException, RecordNotFoundException, and UniqueConstraintViolatedException. When any of these exceptions are thrown, the EIS binding encapsulates the exceptions as corresponding fault business objects like IntegrityConstraintFault, MatchesExceededLimitFault, MissingDataFault, MultipleMatchingRecordsFault, ObjectNotFoundFault, RecordNotFoundFault, and UniqueConstraintFault.

As soon as the EIS binding receives the fault business object, it throws a ServiceBusinessException instead of a ServiceRuntimeException. In global transactions, if a component throws a ServiceRuntimeException, the transaction manager will roll back the global transaction. However, if a component throws a ServiceBusinessException, the transaction manager commits global transaction.

Solution

1. Delete all fault business object binding in the .import file.

```
<methodBinding
      inDataBindingType="com.ibm.xmlns.prod.websphere.j2ca.jdbc.db2admi
      ncustom erbg.Db2adminCustomerBGDataBinding"
      method="createDb2adminCustomerBG"
      outDataBindingType="com.ibm.xmlns.prod.websphere.j2ca.jdbc.db2adm
      incusto merbg.Db2adminCustomerBGDataBinding">
      <faultBinding fault="INTEGRITY_CONSTRAINT_VIOLATION"
      faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
      BindingI    mpl"/>
      <faultBinding fault="MISSING_DATA"
      faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
      BindingI    mpl"/>
      <faultBinding fault="OBJECT_NOTFOUND_EXCEPTION"
      faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
      BindingI    mpl"/>
      <faultBinding fault="UNIQUECONSTRAINT_VIOLATION"
      faultBindingType="com.ibm.j2ca.extension.emd.runtime.WBIFaultData
      BindingI    mpl"/>
      <interaction>
        <properties>
            <functionName>Create</functionName>
        </properties>
      </interaction>
  </methodBinding>
```

2. Define fault handlers to deal with business faults as shown below.

```
DataObject bo = null;
  try {
   bo = (DataObject) locateService_JDBCOutboundInterfacePartner().invoke
("createDb2adminAddressBG",createDb2adminAddressBGInput);
  } catch (ServiceBusinessException e) {
   // TODO Auto-generated catch block
   throw new ServiceRuntimeException(e);
  }
```

   Whenever JDBC adapter outbound has any exception, a ServiceRuntimeException is generated and the global transaction is rolled back. Otherwise, the global transaction is committed.

Sharing a JDBC adapter with one or more modules in an EAR file

To share a JDBC adapter with one or modules within an EAR file by embedded deployment:

1. In the Business Integration perspective of WebSphere Integration Developer, select **File > New > External Service**.
2. Select IBM WebSphere Adapter for JDBC and create an Outbound or Inbound service. Ensure that you add the JDBC drivers to the adapter connector project

in order to connect to various databases. If you want the JDBC adapter to connect to different databases, you can add all the JDBC drivers to the JDBC adapter connector project at once.

3. Select the exiting connector project and create another Outbound or Inbound service. Use the same module that you created earlier for the artifacts.
4. After you create the Inbound or Outbound services, check whether there is only JDBC adapter connector project listed in the **J2EE** area of Dependencies page for the selected module.

For more information, see Sharing a JDBC adapter with one or more modules in an ear file.

Setting the object_key column value in the event table for a composite key

Problem

In the event table which JDBC adapter polls, there are several columns set as the composite primary key. How do you set the object_key column in the event table for the JDBC adapter to poll the event?

Solution

Following is an example to explain the scenario. Here is an Employee table in the database:

```
CREATE TABLE ADMIN.EMPLOYEE ( EMPLOYEEID VARCHAR (10) NOT NULL ,
JOBCODE VARCHAR (10) NOT NULL , EMPLOYEENAME VARCHAR (10) NOT NULL ,
AGE VARCHAR (10) NOT NULL , CONSTRAINT CC1182727951922 PRIMARY KEY (
EMPLOYEEID, JOBCODE) ) ;
```

Both EmployeeID and JobCode are composite keys in the Employee table.

Use the following SQL statement to insert data into the Employee table:

```
INSERT INTO employee (employeeid,jobcode,employeename,age) VALUES
('1','8','Mike','30');
```

Both 1 (EmployeeID) and 8 (JobCode) have a record for the employee "Mike" in the "Employee" table.

When using a trigger to insert a record into the event table, the SQL statement must be as shown below:

```
INSERT INTO wbia_jdbc_eventstore (object_key, object_name,
object_function, event_priority, event_status)
VALUES ('1;8', 'AdminEmployeeBG', 'Create', 1, 0);
```

After running the ESD to generate the business object for the "Employee" table , both the generated key attributes from the business object have the same order as the table definition. The correct order must be "EmployeeID" first and the "JobCode" next.

Thus, when inserting the value of "object_key" into the event table, you have to follow the same order as the business object definition and the table definition. For example, if the "EmployeeID" column has the value "1" and "JobCode" column has the value "8" , then the correct value in "object_key" must be "1;8" and not "8;1".

In addition, you should use the semicolon character (;) as the delimiter to separate each composite key value in the "object_key" column.

Using WebSphere Adapter for JDBC with WebSphere Business Integration Adapter for JDBC for user-defined event query

Problem

When using the XWORLDS_EVENTS event table from WebSphere Business Integration Adapter for JDBC for the user-defined event query instead of the WBIA_JDBC_EventStore event table form WebSphere Adapter for JDBC, an error with the message "field object_function not there" or "Invalid column name" is generated.

Cause

The field "object_function" does not exist in WebSphere Business Integration Adapters event table XWORLDS_EVENTS, and the field "connector_ID" does not exist in the WebSphere Adapter for JDBC event table WBIA_JDBC_EVENTSTORE .

Solution

Here are the migration steps.

1. Create table "WBIA_JDBC_EVENTSTORE" with the connector_ID .

   ```
   CREATE TABLE WBIA_JDBC_EventStore
   (
   event_id INTEGER NOT NULL PRIMARY KEY,
   xid VARCHAR(200),
   object_key VARCHAR(80) NOT NULL,
   object_name VARCHAR(40) NOT NULL,
   object_function VARCHAR(40) NOT NULL,
   event_priority INTEGER NOT NULL,
   event_time TIMESTAMP default CURRENT TIMESTAMP NOT NULL,
   event_status INTEGER NOT NULL,
   connector_ID VARCHAR(40),
   event_comment VARCHAR(100)
   );
   ```

   The "connector_ID" field will be used to filter some events from JDBC adapter instance. The "connector_ID" enables seamless migration for WebSphere Business Integration Adapters customers to JCA where customers are currently taking advantage of the connectorID filtering. This feature will allow the customers to balance load when they have large number of events of the same type.
2. Create and run the below SQL statements.

   user-defined query:

   ```
   select event_id, object_key, object_name, object_verb as object_function ,
   connector_id from xworlds_events where event_status = 0
   ```

   user-defined update:

   ```
    update xworlds_events set event_status= 1 where event_id = ?
   ```

   user-defined delete:

   ```
   delete from xworlds_events where event_id= ?
   ```

Implementing archive events function for inbound service of WebSphere Adapter for JDBC

See Implementing archive events function for inbound service of WebSphere Adapter for JDBC.

JDBC adapter cannot find method for native method during inbound process

Problem

JDBC adapter cannot find method for native method during inbound process.

Cause

JDBC adapter uses native method internally to map each supported operation. The relationship between inbound operation with native method is defined in method bindings of export component properties.

JDBC adapter constructs native method for the Create operation by using the prefix 'emitCreateAfterImage' and the object name, which is retrieved from the adapter event table. If the defined native method in the adapter Export component is emitCreateAfterImageAdminCustomerBG, and the inserted object name into the adapter event table is AdminCUSTOMERBG, then the error is generated because the names of the two native methods do not match each other.

Solution

You can use one of the following ways to fix this issue:

- Insert AdminCustomerBG as object name into the event table. or
- Replace emitCreateAfterImageAdminCustomerBG with emitCreateAfterImageAdminCUSTOMERBG in the properties for adapter Export component.

Null point exception is generated by the driver when retrieving contents from a result set containing a BLOB column

Problem

Null point exception is generated by the driver when the adapter tries to retrieve contents from a result set containing a BLOB column returned by a DB2 stored procedure.

Cause

DB2 JCC driver version is 4.7.85.

Solution

Use DB2 JCC driver 3.50.152.

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:

- Flashes (alerts about the product)
- Technical information including the product information center, manuals, IBM Redbooks®, and whitepapers
- Educational offerings
- Technotes

Recommended fixes

A list of recommended fixes you must apply is available at the following location: http://www.ibm.com/support/docview.wss?fdoc=aimadp&rs=695&uid=swg27010397

Technotes

Technotes provide the most current documentation about 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. The plug-in supports the dynamic trace feature. For information about installing or using IBM Support Assistant, visit this address:

http://www.ibm.com/software/support/isa/

Chapter 10. Reference

Detailed information about business objects, adapter properties (enterprise service discovery properties, resource adapter properties, managed (J2C) connection factory properties, activation specification properties, and interaction specification properties), messages, and related product information is provided for your reference.

Business object information

A business object is a structure that contains application-specific information (metadata) about how the adapter must 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. 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.

© Copyright IBM Corp. 2006, 2009

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
NUMERIC	Decimal Int
INTEGER SMALLINT TINYINT	Int **Note:** For Oracle database, the adapter maps these column types to String.

Table 16. JDBC metadata column type and business object attribute types (continued)

JDBC metadata column type	Business object attribute type
BIGINT	Long Int
TIME	String Time
TIMESTAMP	String DateTime
DATE	String Date Datetime
DECIMAL	Decimal
DOUBLE FLOAT	Double
REAL	Float **Note:** For Oracle database, the adapter maps this column type to Double.
BLOB	hexBinary
CLOB	String
BINARY VARBINARY LONGBINARY	hexBinary
NCHAR NVARCHAR NTEXT	String
TEXT	String
RAW	String **Note:** For Oracle database, the adapter maps this column type to hexBrinary.
MONEY SMALLMONEY	Decimal

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 on page 200.

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 set to `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`
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 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

Table 17. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value
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
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.fffffffff* **Note:** The java.sql.Timestamp.valueOf () method converts the JDBC timestamp format to the Timestamp value and inserts this value into the database. The java.sql.Timestamp.toString() method converts the Timestamp value in the database to string. The adapter uses the Timestamp value present in the database. To learn more about the Timestamp method, see the Sun Web site at http://java.sun.com/j2se/1.5.0/docs/api/ and search for *Timestamp*.	None
DecimalScale	Int	Species the scale of decimal data type. For example, $unscaledVal \times 10^{-scale}$	None
Dummy	Boolean	Indicates a dummy column. If `True`, the Dummy column value is not updated or inserted into the database. Use this application-specific information when you want to configure multiple ForeignKey values on one column.	None

Table 17. Application-specific information for simple attributes (continued)

Parameter	Type	Description	Default value

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 in the business object. Open the business object by using an XML or Text editor to edit the XSD file and make the following two changes:
		1. Remove the `type="string"` added by default to the `<element>` tag for the object attribute.
		2. Add a new `<simpletype>` section before the `</element>` as shown in this example:
		`<xsd:simpleType>`
		`<xsd:restriction base="xsd:string">`
		`<xsd:maxLength value="10"/>`
		`</xsd:restriction>`
		`</xsd:simpleType>`
		
		Save the object definition, and ensure that no validation errors occur in the XSD file after it has been updated.
		
		See the section "Example of a FixedChar parameter in the business object XSD file" following this table.
			`false`

ForeignKey	String	The value of this property depends on whether the parent and 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, the PrimaryKey parameter is set 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 (identity columns are known as *serial columns* in Informix). DB2 supports both sequences and identity columns. Identity columns provide a way for the database to automatically generate a unique numeric value for each row in a table. Identity columns can be defined for DB2 and Microsoft SQL Server, and serial columns can be defined for Informix. The format of this parameter is as follows: `UID=AUTO|`*`Sequence_Name`* If you run the discovery process against a table in either a DB2 or Microsoft SQL Server database, you must manually set the UID (Unique Identifier) attribute to AUTO, for example, `<UID>AUTO</UID>`. **Note:** The requirement to manually set the UID (Unique Identifier) attribute to AUTO is specific to identity columns in DB2 and Microsoft SQL Server. The requirement does not apply to serial columns in Informix. For Informix, the UID attribute for the serial column is generated automatically and will be either <UID>SERIAL</UID> or <UID>SERIAL8</UID>. Like identity columns, sequences are also used to automatically generate numeric values. Consult your database documentation for details on how the database uses sequences and identity columns. For a sequence, set the UID attribute to the name of the sequence. Sequences can be defined for DB2 and Oracle databases. 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 section of an XSD file

```
        <jdbcasi:ColumnName>pkey</jdbcasi:ColumnName>
        <jdbcasi:PrimaryKey>true</jdbcasi:PrimaryKey>
        <jdbcasi:FixedChar>true</jdbcasi:FixedChar>
    </jdbcasi:JDBCAttributeTypeMetadata>
    </appinfo>
```

```
</annotation>
<xsd:simpleType>
   <xsd:restriction base="xsd:string">
   <xsd:maxLength value="10"/>
   </xsd:restriction>
</xsd: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>
<xsd:simpleType>
        <xsd:restriction base="xsd:string">
                <xsd:maxLength value="10"/>
        </xsd:restriction>
</xsd:simpleType>
</element>
```

Application-specific information for attributes that refer to 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 a child business object from being deleted during an Update operation.	None

Table 18. Application-specific information for attributes of type child business object (continued)

Parameter	Type	Description	Default value
Ownership	Boolean	This parameter specifies that a child business object is owned by the parent. If `True`, Create, Update, and Delete operations on the child business object are allowed. If `False`, 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>
```

Another example of an XSD file for single- and multiple-cardinality child business objects

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

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

```
</appinfo>
</annotation>
</element>
```

Application-specific information for operations

The adapter uses application-specific information at the operation level to perform operations, such as those 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.
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 select from a list of types. For information about valid stored procedure types, see “Stored procedure type” on page 41.
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.
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 from 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. 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 returns 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 run time.
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 on page 209.

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 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 is `HREmployee`.

For business objects that do not correspond to database objects, such as business objects for database queries, batch SQL statements, and wrappers, if you give the business object the same name as a table or stored procedure business object, the wizard appends the name with a unique number so that duplicate names do not result.

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.

Table 25. Business object naming conventions (continued)

Element	Naming convention
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. A prefix is not required, and if not specified, no prefix is 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. A number is appended to the name if necessary to differentiate the business object from another business object with the same name. For example, using the prefix Campaign12 for the Customer table in the Sales schema, 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 is added to the business object name. • *QueryBOName* is the value you specified when you configured the business object in the wizard. A number is appended to the name if necessary to differentiate the business object from another business object with the same name.
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 is added to the business object name. • *BatchSQLBOName* is the name you specified when you configured the business object in the wizard. A number is appended to the name if necessary to differentiate the business object from another business object with the same name.
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 is added to the business object name. • *WrapperBOName* is the name you specified when you configured the business object in the wizard. A number is appended to the name if necessary to differentiate the business object from another business object with the same name.

Configuration properties

WebSphere Adapter for JDBC has several categories of configuration properties, which you set with the external service wizard while generating or creating objects and services. You can change the resource adapter, managed connection factory, and activation specification properties after you deploy the application to WebSphere Process Server or WebSphere Enterprise Service Bus.

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 or WebSphere Enterprise Service Bus using WebSphere Integration Developer or the 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 processes the field using its assigned default value, and that default value is displayed on the administrative console. Possible values are **Yes** and **No**. Sometimes a property is required only when another property has a specific value. When this is the case, the table 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.

Row	Explanation
Unit of measure	Specifies how the property is measured, for example in kilobytes or seconds.
Property type	Describes the property type. Valid property types include: • Boolean • String • Integer
Usage	Describes usage conditions or restrictions that might apply to the property. For instance, here is how a restriction would be documented: For Rational® Application Developer for WebSphere Software version 6.40 or earlier, the password: • Must be uppercase • Must be 8 characters in length For versions of Rational Application Developer for WebSphere Software later than 6.40, the password: • Is not case sensitive • Can be up to 40 characters in length. This section lists other properties that affect this property or the properties that are affected by this property and describes the nature of the conditional relationship.
Example	Provides sample property values, for example: "If Language is set to JA (Japanese), code page number is set to `8000`".
Globalized	If a property is globalized, it has national language support, meaning that you can set the value in your national language. Valid values are **Yes** and **No**.
Bidi supported	Indicates whether the property is supported in bidirectional (bidi) processing. Bidirectional processing refers to the task of processing data that contains both right-to-left (Hebrew or Arabic, for example) and left-to-right (a URL or file path, for example) semantic content within the same file. Valid values are **Yes** and **No**.

Connection properties for the wizard

External service connection properties 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 discover objects in the database are used as the initial value for the run time 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 212.

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	Specifies the name of the database.
Database software	The name and version of the database management software used by the adapter.
Database URL	The database URL that is used to connect to the database.
Host name	The host name or IP address of the database server.
JDBC driver class name	The name of the JDBC driver class.
JDBC driver type	The type of JDBC driver to use.
Password	Password for the corresponding user name.
Port number	The port number for connecting to the database instance.
Prefix for business object names	A prefix to be added to the name of the business object.
Server name	The name of the Informix database server to which the adapter connects.
Set Auto Commit on database connection	Specify whether the underlying database connection is in the Auto Commit mode.
User name	Specifies the database user name for connecting to the database.

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. Additional JDBC driver connection properties details

Row	Explanation
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 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 27. Additional JDBC driver connection properties details (continued)

Row	Explanation

Example	The following value of this property specifies a login timeout 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 name details

Row	Explanation
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

Row	Explanation
Required	Yes
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://<Host_Name>/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://<Host_Name>/TEST`
	
	Informix V10
	`jdbc:informix-sql://<Host_Name>/`
	`symaster:INFORMIXSERVER=server`
	
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`
	
	Derby JDBC driver (non-remote)
	`jdbc:derby://<runtime home>/runtimes/bi_v6/derby/`
	`databases/JDBCTEST`
	
	If using a remote test environment on z/OS, use the following value for the Derby database URL:
	
	Derby JDBC driver (remote z/OS test environment)
	`jdbc:db2j:net://<HOST_NAME>:1527//`
	`<remote_derbydb_path>/JDBCTEST</`
	`remote_derbydb_path></HOST_NAME>`

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

Table 31. Host name details (continued)

Row	Explanation
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 class name

This property specifies the name of the JDBC driver class.

Table 32. JDBC driver class name 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	Although the wizard displays the default class name for the JDBC driver type you select, you can type another class name if needed. If you select **Other** as the JDBC driver value, no default is provided, and you need to type the class name. 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)

Password for the corresponding user name.

Table 34. Password details

Row	Explanation
Required	Yes
Default	No default value
Property type	String
Usage	The password associated with the user name entered to connect to the database for the purpose of discovering objects.
Globalized	Yes
Bidi supported	Yes

Port number

This property specifies the port number for the database instance.

Table 35. Port number details

Row	Explanation
Required	Yes
Default	The default value is database-specific, and is initialized by the wizard if you select a named driver in JDBC driver type.
Property type	String
Usage	This is the port number for connecting to the database instance. This property is not enabled if you select `Other` for the JDBC driver type.
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

Row	Explanation
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

Server name

Specifies the default name of the Informix database server to which the adapter connects.

Table 37. Server name details

Row	Explanation
Required	Yes
Default	server
Property type	String
Usage	The value of the server name can be a local or remote server, but must correspond to a valid `dbservername` entry in the `$INFORMIXDIR/etc/sqlhosts` file on the computer running the application. The `dbservername` must begin with a lower-case letter and cannot exceed 128 bytes. It can include any printable characters except uppercase characters, field delimiters (blank space or tab), the newline character, and the hyphen (or minus) symbol.
Globalized	No
Bidi supported	No

Set Auto Commit on database connection

Specify whether the underlying database connection is in the Auto Commit mode. It overrides the default setting in the Sybase database for the transaction mode for the stored procedure.

Table 38. Set auto commit on database connection details

Row	Explanation
Required	Yes, for specific JDBC drivers and fork a certain stored procedure configuration.
Default	False
Property type	Boolean
Usage	Set this value to specify whether the underlying database connection is in Auto Commit mode. If this property is set to `True`, any change to the underlying database will not rollback after the wizard is complete. Setting this property is necessary if you are performing discovery on a Sybase database using the jConnect driver, and the transaction mode setting for stored procedure on the Sybase database is set to "unchained mode" or "Transact-SQL mode". By setting this property to `True`, you override a Sybase transaction mode configuration that prevents the adapter from discovering result sets.
Globalized	Yes
Bidi supported	No

User name (UserName)

This property specifies the database user name for connecting to the database.

Table 39. User name details

Row	Explanation
Required	Yes
Default	No default value
Property type	String
Usage	The user name is the name entered to connect to the database for discovering objects.
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 deprecated:

- LogFileMaxSize
- LogFileName
- LogNumberOfFiles
- TraceFileMaxSize
- TraceFileName
- TraceNumberOfFiles

The following resource adapter properties will be overridden if the same properties are set in the activation specification for inbound processing or managed connection factory for outbound processing.

- DatabaseVendor
- PingQuery
- QueryTimeOut
- ReturnDummyBOForSP

The BONameSpace 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. For information about how to read the property details table, see "Guide to information about properties" on page 212.

Table 40. Resource adapter properties for the Adapter for JDBC

Property name		
In the wizard	**In the administrative console**	**Description**
Adapter ID	AdapterID	Identifies the adapter instance for PMI events and for logging and tracing.
Database vendor	DatabaseVendor	The type of database that the adapter uses for special processing.

Table 40. Resource adapter properties for the Adapter for JDBC (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
Disguise user data as "XXX" in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing `X` strings instead of user data in the log and trace files.
Query timeout (seconds)	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)	LogFileSize	Deprecated
(Not available)	LogFilename	Deprecated
(Not available)	LogNumberOfFiles	Deprecated
SQL query to verify the connection	PingQuery	The SQL query used to test the reliability of the connection to the database.
(Not available)	TraceFileSize	Deprecated
(Not available)	TraceFileName	Deprecated
(Not available)	TraceNumberOfFiles	Deprecated

Adapter ID (AdapterID)

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

Table 41. Adapter ID details

Required	Yes
Default	`001`
Property type	String

Table 41. Adapter ID details (continued)

Usage	This property identifies the adapter instance in the log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, `JDBCRA`, to form the component name used by the Log and Trace Analyzer tool. For example, if the adapter ID property is set to `001`, the component ID is `JDBCRA001`. If you run multiple instances of the same adapter, ensure that the first seven characters of the adapter ID property are unique for each instance so that you can correlate the log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate the log and trace information to a particular instance of an adapter. For example, when you set the adapter ID property of two instances of WebSphere Adapter for JDBC to `001` and `002`. The component IDs for those instances, `JDBCRA001` and `JDBCRA002`, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. If you set the adapter ID properties of two instances to `Instance01` and `Instance02`, you will not be able to examine the log and trace information for each adapter instance because the component ID for both instances is truncated to `JDBCRAInstanc`. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, the value can be set both at the resource adapter level and the managed connection factory level. After you use the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, ensure that you set them consistently, to prevent inconsistent marking of the log and trace entries.
Globalized	Yes
Bidi supported	No

Database vendor (DatabaseVendor)

This property specifies the type of database that is used. The type is determined by the database vendor name.

Table 42. Database vendor details

Required	Yes

Possible values	`DB2`
	`Informix`
	`MSSQLServer`
	`Oracle`
	`Others`

Default	No default value

Table 42. Database vendor details (continued)

Property type	String
Usage	Some SQL statements require special processing, which varies by database type. For example, the Struct and Array data types in Oracle require special processing. This property specifies the RDBMS that is used, which determines the database type. For other databases, 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

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

This property specifies whether to replace user data in log and trace files with a string of `X`'s to prevent unauthorized disclosure of potentially sensitive data.

Table 43. Disguise user data as "XXX" in log and trace files details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	If you set this property to `True`, the adapter replaces user data with a string of `X`'s when writing to log and trace files. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, the value can be set both at the resource adapter level and the managed connection factory level. After you use the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, ensure that you set them consistently, to prevent inconsistent marking of the log and trace entries.
Globalized	No
Bidi supported	No

Enable high availability support (enableHASupport)

Do not change this property. It must be set to `true`.

Query timeout (seconds) (QueryTimeOut)

This property specifies the maximum number of seconds a query can take to run all SQL statements.

Table 44. 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	No
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 45. 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 the ReturnDummyBOForSP property is set to `True`, a dummy business object is created and the returned parameters (including the out and inout parameters) from the stored procedure are populated into the corresponding attributes
Globalized	No
Bidi supported	No

SQL query to verify the connection (PingQuery)

This property specifies the SQL query that is used to test the reliability of the connection to the database.

Table 46. Ping query details

Required	No
Property type	String
Default	No default value

Table 46. Ping query details (continued)

Usage	This property contains the SQL query statement that you want to run to determine whether the adapter can connect to the database. The adapter runs the ping query every time it receives a SQLException exception while performing the outbound operation. The adapter does not try to recover the connection. If the ping query indicates that the connection to the database is no longer valid, the adapter notifies the container. It is the responsibility of the connection pool manager to remove the stale connection from the pool, which allows subsequent outbound requests to be processed.
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 through the external service wizard during adapter configuration. You can change them before deployment by 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 describes the managed connection factory properties. A complete description of each property is provided in the sections that follow the table. For information about the sections that follow, see "Guide to information about properties" on page 212.

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 47. 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 PMI events and for logging and tracing.
Additional JDBC driver connection properties [name:value;name:value]	JDBCDriverConnectionProperties	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.
Database connection information	ConnectionType	Specifies how the adapter establishes connection to the database.
Data source JNDI name	DataSourceJNDIName	Deprecated.
Database URL	DatabaseURL	The database URL that is used to connect to the database.

Table 47. Managed connection factory properties for Adapter for JDBC (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
Database vendor	DatabaseVendor	The type of database that the adapter uses for special processing.
(Not available)	ErrorOnEmptyResultSet	Specifies whether to generate an exception when no records are found.
Disguise user data as "XXX" in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing `X` strings instead of user data in the log and trace files.
JDBC driver class	JDBCDriverClass	The class name of the JDBC driver that is used to connect to the database.
Maximum retries on connection failure	ConnectionRetryLimit	Specifies the maximum number of times the adapter tries to reestablish an outbound connection to the EIS.
Password	Password	Password for the corresponding user name
Connection pool DataSource JNDI name	PoolDataSourceJNDIName	The JNDI name of the connection pool data source to be used to establish a connection to the database.
Query timeout (seconds)	QueryTimeOut	The maximum number of seconds a query can take for all SQL statements.
Connection retry interval (in milliseconds)	ConnectionRetryInterval	Specifies the time interval between attempts to reconnect to the EIS if the connection fails.
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 verify the connection	PingQuery	The SQL query used to test the reliability of the connection to the database.
User name	UserName	The database user name.
XA DataSource name	XADataSourceName	The name of the XA data source to use to establish a connection to the database for XA (distributed) transactions.
XA database name	XADatabaseName	The database name used for the XA connection.
XA DataSource JNDI name	XADataSourceJNDIName	The JNDI name of the XA data source used to establish a connection to the database.

Adapter ID (AdapterID)

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

Table 48. Adapter ID details

Required	Yes
Default	`001`
Property type	String

Table 48. Adapter ID details (continued)

Usage	This property identifies the adapter instance in the log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, `JDBCRA`, to form the component name used by the Log and Trace Analyzer tool. For example, if the adapter ID property is set to `001`, the component ID is `JDBCRA001`. If you run multiple instances of the same adapter, ensure that the first seven characters of the adapter ID property are unique for each instance so that you can correlate the log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate the log and trace information to a particular instance of an adapter. For example, when you set the adapter ID property of two instances of WebSphere Adapter for JDBC to `001` and `002`. The component IDs for those instances, `JDBCRA001` and `JDBCRA002`, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. If you set the adapter ID properties of two instances to `Instance01` and `Instance02`, you will not be able to examine the log and trace information for each adapter instance because the component ID for both instances is truncated to `JDBCRAInstanc`. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, the value can be set both at the resource adapter level and the managed connection factory level. After you use the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, ensure that you set them consistently, to prevent inconsistent marking of the log and trace entries.
Globalized	Yes
Bidi supported	No

Additional JDBC driver connection properties [name:value;name:value] (JDBCDriverConnectionProperties)

This property contains additional information for connecting to the database using the JDBC driver.

Table 49. Additional JDBC driver connection properties details

Row	**Explanation**
Required	No
Possible values	Database connection properties are database-specific.
Default	No default value
Property type	String

Table 49. Additional JDBC driver connection properties details (continued)

Row	Explanation
Usage	These connection properties are used in addition to the UserName and Password properties 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 timeout 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 50. 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

Database URL (DatabaseURL)

This property specifies the JDBC driver-specific URL for creating a connection to the database.

Table 51. Database URL details

Required	Yes, when the connection is of the type LocalConnectionProps or ConnectionProps.
Default	No default value
Property type	String
Usage	In the external service wizard, compose the database URL by entering 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. If you specify the host name as an IP address in IPv6 format, enclose the IP address in square brackets ([]).

Table 51. 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 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 the type of database that is used. The type is determined by the database vendor name.

Table 52. Database vendor details

Required	Yes

Possible values	`DB2`
	`Informix`
	`MSSQLServer`
	`Oracle`
	`Others`

Default	No default value
Property type	String
Usage	Some SQL statements require special processing, which varies by database type. For example, the Struct and Array data types in Oracle require special processing. This property specifies the RDBMS that is used, which determines the database type. For other databases, 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

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

This property specifies whether to replace user data in log and trace files with a string of `X`'s to prevent unauthorized disclosure of potentially sensitive data.

Table 53. Disguise user data as "XXX" in log and trace files details

Required	No

Table 53. Disguise user data as "XXX" in log and trace files details (continued)

Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	If you set this property to `True`, the adapter replaces user data with a string of `X`'s when writing to log and trace files. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, the value can be set both at the resource adapter level and the managed connection factory level. After you use the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, ensure that you set them consistently, to prevent inconsistent marking of the log and trace entries.
Globalized	No
Bidi supported	No

Throw exception when no records are found (ErrorOnEmptyResultSet)

This property specifies whether the ErrorOnEmptyResultSet property is set for the RecordNotFoundException property on the RetriveAll operation.

Table 54. Throw exception when no records are found details

Required	No
Possible values	`True` `False`
Default	`True`
Property type	Boolean
Usage	If you set this property to `False`, the RetrieveAll operation does not generate a RecordNotFoundException exception when no records are found.
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 55. JDBC driver class details

Row	Explanation
Required	Yes, when the connection is of the type LocalConnectionProps or ConnectionProps.
Possible values	Values are database-specific.
Default	No default value

Table 55. JDBC driver class details (continued)

Property type	String
Usage	In the external service wizard, the JDBC driver class is displayed 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 type the database 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.

Examples	Values for the JDBC driver class is displayed in both the external service wizard and the administrative console. The following examples depict the JDBC driver class properties in both the external service wizard and the administrative console.**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 56. Password details

Required	No. For inbound processing, if you set the Authentication alias or the DataSourceJNDIName, the password is not mandatory. However, if you set the DataSourceJNDIName, and the Password field, the value specified for the Password takes precedence. For outbound processing, if you set the Authentication alias, XADataSourceJNDIName or PoolDataSourceJNDIName property, the password is not mandatory. However, if you set the XADataSourceJNDIName or PoolDataSourceJNDIName property, and the Password field, the value specified for the Password takes precedence.
Default	No default value
Property type	String
Usage	For inbound processing, setting this property overrides the password specified for the data source on the server using the Authentication alias or DataSourceJNDIName property. For outbound processing, setting this property overrides the password specified for the data source on the server using the Authentication alias, or XADataSourceJNDIName or PoolDataSourceJNDIName property. If you specify JAAS as the security credential, the authentication alias will override this property.
Globalized	Yes
Bidi supported	Yes

Query timeout (seconds) (QueryTimeOut)

This property specifies the maximum number of seconds a query can take to run all SQL statements.

Table 57. 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	No
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 58. 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 the ReturnDummyBOForSP property is set to `True`, a dummy business object is created and the returned parameters (including the out and inout parameters) from the stored procedure are populated into the corresponding attributes
Globalized	No
Bidi supported	No

SQL query to verify the connection (PingQuery)

This property specifies the SQL query that is used to test the reliability of the connection to the database.

Table 59. 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. The adapter runs the ping query every time it receives a SQLException exception while performing the outbound operation. The adapter does not try to recover the connection. If the ping query indicates that the connection to the database is no longer valid, the adapter notifies the container. It is the responsibility of the connection pool manager to remove the stale connection from the pool, which allows subsequent outbound requests to be processed.
Globalized	No
Bidi supported	No

User name (UserName)

This property specifies the database user name that is used to access the database.

Table 60. User name details

Required	No. For inbound processing, if you set the Authentication alias or the DataSourceJNDIName, the User name property is not mandatory. However, if you set the DataSourceJNDIName, and User name field, the value specified for the User name takes precedence. For outbound processing, if you set the Authentication alias, XADataSourceJNDIName or PoolDataSourceJNDIName property, the User name is not mandatory. However, if you set the XADataSourceJNDIName or PoolDataSourceJNDIName property, and the User name field, the value specified for the User name takes precedence.
Default	No default value
Property type	String
Usage	For inbound processing setting this property overrides the user name specified for the data source on the server using the DataSourceJNDIName property or the Authentication alias. For outbound processing, setting this property overrides the user name specified for the data source on the server using the XADataSourceJNDIName property, PoolDataSourceJNDIName property or the Authentication alias. If you specify JAAS as the security credential, the authentication alias will override this property.
Globalized	Yes
Bidi supported	Yes

XA DataSource name (XADataSourceName)

This property specifies the name of the XA data source used to establish a connection to the database for XA (distributed) transactions.

Table 61. 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 with the XADatabaseName property. To make an XA connection to an Oracle database, this property is used, but the XADatabaseName property is not used.

Table 61. XA data source name details (continued)

Examples	A typical value for an Oracle database:
	
	`oracle.jdbc.xa.client.OracleXADataSource`
	
	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`

Globalized	No
Bidi supported	No

XA database name (XADatabaseName)

This property specifies the name of the database used for the XA connection.

Table 62. 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.
Globalized	Yes
Bidi supported	Yes

Maximum retries on connection failure (connectionRetryLimit)

This property specifies the number of times the adapter will attempt to reestablish a connection to the EIS, when the adapter encounters an error related to the outbound connection.

Table 63. Maximum retries in case of system connection failure details

Required	No
Possible values	Integer equal to and greater than zero
Default	0
Property type	Integer

Table 63. Maximum retries in case of system connection failure details (continued)

Usage	When the adapter encounters an error related to the outbound connection, it attempts to reestablish a physical connection (when physical connection is not established) for the number of times specified for this property with a time delay specified in the property "Connection retry interval (in milliseconds) (ConnectionRetryInterval)." If the value is set to 0, the adapter does not validate the connection to the EIS and it executes the outbound operation. If the EIS connection is invalid, the outbound operation fails. The adapter does not attempt to reconnect to the EIS. If the value is set to greater than 0, then during each request the adapter validates if the connection to the EIS is active. • If the connection is valid, the operation is completed. • If connection is invalid, the adapter terminates the current managed connection and a new managed connection is created (new physical connection). If the adapter is successful in reestablishing the connection to the EIS, the outbound operation is completed otherwise the adapter generates the ResourceException after trying to reconnect for the specified number of times.
Globalized	No
Bidi supported	No

Connection retry interval (in milliseconds) (ConnectionRetryInterval)

This property specifies the time interval between attempts to reconnect to the EIS if the connection fails.

Table 64. Retry interval if connection failure details

Required	No
Possible values	Integer equal to and greater than zero
Default	60000
Unit of measure	Milliseconds
Property type	Integer
Usage	When the adapter encounters an error while establishing a connection to the EIS, this property specifies the time interval the adapter waits between attempts to reestablish a connection. By default, this property is disabled and it is enabled only when the value of the "Maximum retries on connection failure (connectionRetryLimit)" on page 235 property is set to greater than zero.
Globalized	No
Bidi supported	No

XA DataSource JNDI name (XADataSourceJNDIName)

This property specifies the JNDI name of the XA data source used to establish a connection to the database.

Table 65. XA Data source JNDI name details

Required	No
Default	No default value
Property type	String
Usage	Use this property to specify the JNDI name of a XA data source in WebSphere Process Server or WebSphere Enterprise Service Bus that specifies connection information for the target database. If the “Database connection information (ConnectionType)” is set to "XADataSourceJNDI" the adapter uses this property to establish the connection to the database. To improve the performance of outbound operations, specify the name of a data source that is enabled for prepared statement caching. If the other valid authentication properties are also set, they override authentication properties in the data source.
Globalized	Yes
Bidi supported	No

Database connection information (ConnectionType)

This property specifies how the adapter establishes connection to the database.

Table 66. Database connection information

Required	Yes
Possible values	XADataSourceJNDI, XAConnectionProps, PoolDataSourceJNDI, or LocalConnectionProps
Default	No default value
Property type	String

Table 66. Database connection information (continued)

Usage	This property specifies how the adapter establishes database connection at run time. This property can have the following values: • XADataSourceJNDI- Indicates that the database connection is established using the XADataSourceJNDIName property that corresponds to the predefined XA data source. • XAConnectionProps - Indicates that the database connection is established using the XADataSourceName and DatabaseURL or XADatabaseName properties. • PoolDataSourceJNDI - Indicates that the database connection is established using the poolDataSourceJNDIName property that corresponds to the predefined data source. • LocalConnectionProps - Indicates that the database connection is established using the DatabaseURL and JDBCDriverClass properties. For a new application, this property is automatically set by external service wizard. For a migrated application, this property is set during the migration process according to the ManagedConnectionFactory properties. • If the DataSourceJNDIName property is set, the value of this property is set to XADataSourceJNDI. • If the XADataSourceName is set, the value of this property is set to XAConnectionProps. • Otherwise, the value of this property is set to LocalConnectionProps. If this property is not set, adapter uses backward compatibility mode to establish database connection. In the backward compatibility mode, 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. 2. If the DataSourceJNDIName property is not set, and the XADataSourceName and XADatabaseName properties are set, the adapter uses them 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.
Globalized	No
Bidi supported	No

Connection pool DataSource JNDI name (PoolDataSourceJNDIName)

The JNDI name of the connection pool data source to be used to establish a connection to the database.

Table 67. Connection pool data source

Required	No
Default	No default value
Property type	String
Usage	Use this property to specify the JNDI name of a connection pool data source in WebSphere Process Server or WebSphere Enterprise Service Bus that specifies connection information for the target database. If the “Database connection information (ConnectionType)” on page 237 property is set to "PoolDataSourceJNDI" the adapter use this property to establish the connection to the database. To improve the performance of outbound operations, specify the name of a data source that is enabled for prepared statement caching. If the other valid authentication properties are also set, they override authentication properties in the data source.
Globalized	No
Bidi supported	No

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 68 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 212.

Table 68. 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 69. Maximum records for RetrieveAll operation details

Required	Yes
Default	100

Table 69. Maximum records for RetrieveAll operation details (continued)

Usage	Use this property to control the number of records returned by the RetrieveAll operation. • If the value -1, the RetrieveAll operation returns all records matching the query. The value for this property is set to -1 internally when you select the **Return all records for RetrieveAll operation** check box. • If the value is zero or less than zero except -1, the adapter generates the exception MatchesExceededLimitException. • If the value is greater than zero and the number of matches in the database exceeds the value of this property, the adapter generates the exception MatchesExceededLimitException. If the RetrieveAll operation does not return all the records, increase this value. For example, if you set the value to 50 and the table contains 100 records, the adapter generates the exception MatchesExceededLimitException. • If the value is greater than zero and the number of matches in the database is lesser than the value of this property, the RetrieveAll operation returns all records. For example, if you set the value to 50 and the table contains 25 records, the RetrieveAll operation returns all the 25 records.
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 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 processes the field using its assigned default value, and that default value is displayed on the administrative console. Possible values are **Yes** and **No**. Sometimes a property is required only when another property has a specific value. When this is the case, the table 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: • Boolean • String • Integer
Usage	Describes usage conditions or restrictions that might apply to the property. For instance, here is how a restriction would be documented: For Rational Application Developer for WebSphere Software version 6.40 or earlier, the password: • Must be uppercase • Must be 8 characters in length For versions of Rational Application Developer for WebSphere Software later than 6.40, the password: • Is not case sensitive • Can be up to 40 characters in length. This section lists other properties that affect this property or the properties that are affected by this property and describes the nature of the conditional relationship.
Example	Provides sample property values, for example: "If Language is set to JA (Japanese), code page number is set to `8000`".

Row	Explanation
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 refers to the task of processing data that contains both right-to-left (Hebrew or Arabic, for example) and left-to-right (a URL or file path, for example) semantic content within the same file. Valid values are **Yes** and **No**.

Connection properties for the wizard

External service connection properties 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 discover objects in the database are used as the initial value for the run time 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 212.

Table 70. 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	Specifies the name of the database.
Database software	The name and version of the database management software used by the adapter.
Database URL	The database URL that is used to connect to the database.
Host name	The host name or IP address of the database server.
JDBC driver class name	The name of the JDBC driver class.
JDBC driver type	The type of JDBC driver to use.
Password	Password for the corresponding user name.
Port number	The port number for connecting to the database instance.
Prefix for business object names	A prefix to be added to the name of the business object.
Server name	The name of the Informix database server to which the adapter connects.

Table 70. Connection properties for the external service wizard (continued)

Property name in the wizard	Description
Set Auto Commit on database connection	Specify whether the underlying database connection is in the Auto Commit mode.
User name	Specifies the database user name for connecting to the database.

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 71. Additional JDBC driver connection properties details

Row	Explanation
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 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 timeout 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 72. Database name details

Row	Explanation
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

Table 72. Database name details (continued)

Row	Explanation
Bidi supported	Yes

Database software

This property specifies the database management software that manages the database that the adapter will access.

Table 73. 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 74. Database URL details

Row	Explanation
Required	Yes
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 74. Database URL details (continued)

Row	Explanation

Examples	The following are typical values for common database servers.
	DB2 universal (type 4) JDBC driver
	`jdbc:db2://<Host_Name>/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://<Host_Name>/TEST`
	Informix V10
	`jdbc:informix-sql://<Host_Name>/ symaster:INFORMIXSERVER=server`
	Oracle V10
	`jdbc:oracle:thin:@9.26.248.148:1521:dev`
	Derby JDBC driver (non-remote)
	`jdbc:derby://<runtime home>/runtimes/bi_v6/derby/ databases/JDBCTEST`
	If using a remote test environment on z/OS, use the following value for the Derby database URL:
	Derby JDBC driver (remote z/OS test environment)
	`jdbc:db2j:net://<HOST_NAME>:1527// <remote_derbydb_path>/JDBCTEST</ remote_derbydb_path></HOST_NAME>`

Globalized	Yes
Bidi supported	Yes

Host name

This property specifies the host name or IP address of the database server.

Table 75. 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 class name

This property specifies the name of the JDBC driver class.

Table 76. JDBC driver class name 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	Although the wizard displays the default class name for the JDBC driver type you select, you can type another class name if needed. If you select **Other** as the JDBC driver value, no default is provided, and you need to type the class name. 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 77. 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)

Password for the corresponding user name.

Table 78. Password details

Row	Explanation
Required	Yes
Default	No default value

Table 78. Password details (continued)

Row	Explanation
Property type	String
Usage	The password associated with the user name entered to connect to the database for the purpose of discovering objects.
Globalized	Yes
Bidi supported	Yes

Port number

This property specifies the port number for the database instance.

Table 79. Port number details

Row	Explanation
Required	Yes
Default	The default value is database-specific, and is initialized by the wizard if you select a named driver in JDBC driver type.
Property type	String
Usage	This is the port number for connecting to the database instance. This property is not enabled if you select `Other` for the JDBC driver type.
Globalized	Yes
Bidi supported	No

Prefix for business object names

The prefix to be added to the name of the business object.

Table 80. Prefix details

Row	Explanation
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

Server name

Specifies the default name of the Informix database server to which the adapter connects.

Table 81. Server name details

Row	Explanation
Required	Yes
Default	server
Property type	String
Usage	The value of the server name can be a local or remote server, but must correspond to a valid `dbservername` entry in the `$INFORMIXDIR/etc/sqlhosts` file on the computer running the application. The `dbservername` must begin with a lower-case letter and cannot exceed 128 bytes. It can include any printable characters except uppercase characters, field delimiters (blank space or tab), the newline character, and the hyphen (or minus) symbol.
Globalized	No
Bidi supported	No

Set Auto Commit on database connection

Specify whether the underlying database connection is in the Auto Commit mode. It overrides the default setting in the Sybase database for the transaction mode for the stored procedure.

Table 82. Set auto commit on database connection details

Row	Explanation
Required	Yes, for specific JDBC drivers and fork a certain stored procedure configuration.
Default	False
Property type	Boolean
Usage	Set this value to specify whether the underlying database connection is in Auto Commit mode. If this property is set to `True`, any change to the underlying database will not rollback after the wizard is complete. Setting this property is necessary if you are performing discovery on a Sybase database using the jConnect driver, and the transaction mode setting for stored procedure on the Sybase database is set to "unchained mode" or "Transact-SQL mode". By setting this property to `True`, you override a Sybase transaction mode configuration that prevents the adapter from discovering result sets.
Globalized	Yes
Bidi supported	No

User name (UserName)

This property specifies the database user name for connecting to the database.

Table 83. User name details

Row	Explanation
Required	Yes
Default	No default value
Property type	String
Usage	The user name is the name entered to connect to the database for discovering objects.
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 deprecated:

- LogFileMaxSize
- LogFileName
- LogNumberOfFiles
- TraceFileMaxSize
- TraceFileName
- TraceNumberOfFiles

The following resource adapter properties will be overridden if the same properties are set in the activation specification for inbound processing or managed connection factory for outbound processing.

- DatabaseVendor
- PingQuery
- QueryTimeOut
- ReturnDummyBOForSP

The BONameSpace 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. For information about how to read the property details table, see "Guide to information about properties" on page 212.

Table 84. Resource adapter properties for the Adapter for JDBC

Property name		
In the wizard	**In the administrative console**	**Description**
Adapter ID	AdapterID	Identifies the adapter instance for PMI events and for logging and tracing.
Database vendor	DatabaseVendor	The type of database that the adapter uses for special processing.

Table 84. Resource adapter properties for the Adapter for JDBC (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
Disguise user data as "XXX" in log and trace files	HideConfidentialTrace	Specifies whether to disguise potentially sensitive information by writing `X` strings instead of user data in the log and trace files.
Query timeout (seconds)	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)	LogFileSize	Deprecated
(Not available)	LogFilename	Deprecated
(Not available)	LogNumberOfFiles	Deprecated
SQL query to verify the connection	PingQuery	The SQL query used to test the reliability of the connection to the database.
(Not available)	TraceFileSize	Deprecated
(Not available)	TraceFileName	Deprecated
(Not available)	TraceNumberOfFiles	Deprecated

Adapter ID (AdapterID)

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

Table 85. Adapter ID details

Required	Yes
Default	`001`
Property type	String

Table 85. Adapter ID details (continued)

Usage	This property identifies the adapter instance in the log and trace files, and also helps identify the adapter instance while monitoring adapters. The adapter ID is used with an adapter-specific identifier, `JDBCRA`, to form the component name used by the Log and Trace Analyzer tool. For example, if the adapter ID property is set to `001`, the component ID is `JDBCRA001`. If you run multiple instances of the same adapter, ensure that the first seven characters of the adapter ID property are unique for each instance so that you can correlate the log and trace information to a particular adapter instance. By making the first seven characters of an adapter ID property unique, the component ID for multiple instances of that adapter is also unique, allowing you to correlate the log and trace information to a particular instance of an adapter. For example, when you set the adapter ID property of two instances of WebSphere Adapter for JDBC to `001` and `002`. The component IDs for those instances, `JDBCRA001` and `JDBCRA002`, are short enough to remain unique, enabling you to distinguish them as separate adapter instances. However, instances with longer adapter ID properties cannot be distinguished from each other. If you set the adapter ID properties of two instances to `Instance01` and `Instance02`, you will not be able to examine the log and trace information for each adapter instance because the component ID for both instances is truncated to `JDBCRAInstanc`. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, the value can be set both at the resource adapter level and the managed connection factory level. After you use the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, ensure that you set them consistently, to prevent inconsistent marking of the log and trace entries.
Globalized	Yes
Bidi supported	No

Database vendor (DatabaseVendor)

This property specifies the type of database that is used. The type is determined by the database vendor name.

Table 86. Database vendor details

Required	Yes

Possible values	`DB2`
	`Informix`
	`MSSQLServer`
	`Oracle`
	`Others`

Default	No default value

Table 86. Database vendor details (continued)

Property type	String
Usage	Some SQL statements require special processing, which varies by database type. For example, the Struct and Array data types in Oracle require special processing. This property specifies the RDBMS that is used, which determines the database type. For other databases, 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

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

This property specifies whether to replace user data in log and trace files with a string of `X`'s to prevent unauthorized disclosure of potentially sensitive data.

Table 87. Disguise user data as "XXX" in log and trace files details

Required	No
Possible values	`True` `False`
Default	`False`
Property type	Boolean
Usage	If you set this property to `True`, the adapter replaces user data with a string of `X`'s when writing to log and trace files. For inbound processing, the value of this property is set at the resource adapter level. For outbound processing, the value can be set both at the resource adapter level and the managed connection factory level. After you use the external service wizard to configure the adapter for outbound processing, you can set the resource adapter and managed connection factory properties independently. If you use the WebSphere Integration Developer assembly editor or the administrative console to reset these properties, ensure that you set them consistently, to prevent inconsistent marking of the log and trace entries.
Globalized	No
Bidi supported	No

Enable high availability support (enableHASupport)

Do not change this property. It must be set to `true`.

Query timeout (seconds) (QueryTimeOut)

This property specifies the maximum number of seconds a query can take to run all SQL statements.

Table 88. 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	No
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 89. 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 the ReturnDummyBOForSP property is set to `True`, a dummy business object is created and the returned parameters (including the out and inout parameters) from the stored procedure are populated into the corresponding attributes
Globalized	No
Bidi supported	No

SQL query to verify the connection (PingQuery)

This property specifies the SQL query that is used to test the reliability of the connection to the database.

Table 90. Ping query details

Required	No
Property type	String
Default	No default value

Table 90. Ping query details (continued)

Usage	This property contains the SQL query statement that you want to run to determine whether the adapter can connect to the database. The adapter runs the ping query every time it receives a SQLException exception while performing the outbound operation. The adapter does not try to recover the connection. If the ping query indicates that the connection to the database is no longer valid, the adapter notifies the container. It is the responsibility of the connection pool manager to remove the stale connection from the pool, which allows subsequent outbound requests to be processed.
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 212.

Table 91. Activation specification properties for Adapter for JDBC

Property name		
In the wizard	**In the administrative console**	**Description**
Adapter Instance for event filtering	AdapterInstanceEventFilter	Identifier that determines whether this adapter instance processes specific events in the event store.
Additional JDBC driver connection properties	JDBCDriverConnectionProperties	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	BONameSpace	The namespace for the business object definitions.
User-defined 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.
User-defined event query	CustomEventQuery	The name of the query, stored procedure, or stored function that performs the polling for events.

Table 91. Activation specification properties for Adapter for JDBC (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
User-defined 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.
Database connection information	ConnectionType	Specifies how the adapter establishes connection to the database.
Data source JNDI name	DataSourceJNDIName	The name of the JNDI data source used 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.
Retry limit for failed events	FailedEventRetryLimit	The number of times the adapter attempts to redeliver an event before marking the event as failed.
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.
Minimum connections	MinimumConnections	The minimum number of connections that the adapter can use for inbound event delivery.
Maximum number of retries in case of system connection failure	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 the adapter delivers to the export during each poll period.

Table 91. Activation specification properties for Adapter for JDBC (continued)

Property name		
In the wizard	**In the administrative console**	**Description**
Query time out	QueryTimeOut	The maximum number of seconds a query can take for all SQL statements.
Retry EIS connection on startup	RetryConnectionOnStartup	Controls whether the adapter retries the connection to the database if it cannot connect at startup.
Time between retries in case of system connection failure (milliseconds)	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 the reliability of the 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.
(Not available)	performTrimOnObjectKeyValue	Specifies whether to trim spaces for key-value pairs.
(Not available)	nameValuePairDelimiter	Specifies the delimiter to separate key-value pair.
(Not available)	valueDelimeter	Specifies the delimiter to separate the key and value in a key-value pair.
(Not available)	unsetValueKeyWord	Specifies the unset value in the object_key field.
(Not available)	nullValueKeyWord	Specifies the null value in the object_key field.

Adapter Instance for event filtering (AdapterInstanceEventFilter)

This property controls whether the adapter instance processes specific events in the event store.

Table 92. Adapter Instance for event filtering details

Required	No
Default	`null`
Property type	String

Table 92. Adapter Instance for event filtering details (continued)

Usage	This property helps you migrate from WebSphere Business Integration Adapter for JDBC to WebSphere Adapter for JDBC. WebSphere Business Integration Adapter for JDBC allows you to perform load balancing on high-volume event types by allowing multiple adapter instances to process events of the same type. When load balancing is not required, a single adapter instance processes all events of a given type. This property is to enable seamless migration for WBIA customers to JCA for customers who are currently taking advantage of the connectorID filtering. WebSphere Adapter for JDBC typically does not require load balancing in this way, but supports it so that you can migrate without modifying the database triggers or other mechanisms that write events to the event store. The AdapterInstanceEventFilter property corresponds to the ConnectorID property of the WebSphere Business Integration Adapter for JDBC. To use this feature, the database triggers or other mechanisms that create events in the event store must assign the appropriate value to the connector_ID column. Table 93 shows the interaction between the AdapterInstanceEventFilter property and the value in the connector_ID column in the event store. This property applies only to standard event processing. For custom event processing, the custom event query must perform necessary filtering. If the EventTypeFilter and AdapterInstanceEventFilter properties are both set, the adapter processes only events that meet both criteria. That is, it processes only those events whose type is specified in the EventTypeFilter property and whose connector_ID column matches the AdapterInstanceEventFilter property.
Example	See Table 93.
Globalized	Yes
Bidi supported	Yes

Table 93. Interaction of the AdapterInstanceEventFilter property with the connector_ID column in the event store

AdapterInstanceEventFilter property	**connector_ID column of an event**	**Result**
`null`	`null`	The adapter processes the event
`null`	`Instance1`	The adapter processes the event, because the connector_ID column is not checked
`Instance1`	`Instance1`	The adapter processes the event
`Instance1`	`Instance2`	The adapter does not process the event, because the instance IDs do not match
`Instance1`	`null`	The adapter does not process the event, because the instance IDs do not match

Additional JDBC driver connection properties [name:value;name:value] (JDBCDriverConnectionProperties)

This property contains additional information for connecting to the database using the JDBC driver.

Table 94. Additional JDBC driver connection properties details

Row	Explanation
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 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 timeout 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 (BONameSpace)

This property specifies the namespace for the business object definitions.

Table 95. Business object namespace property characteristics

Required	No
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

User-defined 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 96. Custom delete query details

Required	No

Table 96. Custom delete query details (continued)

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`. **Note:** During Bidi transformation, only the string within the quotes after the WHERE clause are converted for query with conditions. For example, in the query `select * from customer where customerid='john'`, only the string 'john' is converted.
Globalized	Yes
Bidi supported	Yes

User-defined 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 97. 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`.

Examples	In the following example, the custom event query runs a 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_TABLE`
	`where 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(?)`
	
	During Bidi transformation, only the string within the quotes after the WHERE clause are converted for query with conditions. For example, in the query `select * from customer where customerid='john'`, only the string 'john' is converted.

Globalized	Yes
Bidi supported	Yes

User-defined 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 98. 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`. **Note:** During Bidi transformation, only the string within the quotes after the WHERE clause are converted for query with conditions. For example, in the query `select * from customer where customerid='john'`, only the string 'john' is converted.
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 99. Data source JNDI name details

Required	No
Default	No default value
Property type	String

Table 99. Data source JNDI name details (continued)

Usage	Use this property to specify the JNDI name of a data source in WebSphere Process Server or 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 UserName and Password properties are also set, they override the user name and password in the data source. 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 – Is used instead of saving user name and password in adapter properties – Is 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 – Is used instead of specifying the value of each managed connection factory or activation specification property in the wizard – Is 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 100. Database URL details

Required	Yes, when the connection is of the type LocalConnectionProps or ConnectionProps.
Default	No default value
Property type	String
Usage	In the external service wizard, compose the database URL by entering 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. If you specify the host name as an IP address in IPv6 format, enclose the IP address in square brackets ([]).

Table 100. 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 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 the type of database that is used. The type is determined by the database vendor name.

Table 101. Database vendor details

Required	Yes

Possible values	`DB2`
	`Informix`
	`MSSQLServer`
	`Oracle`
	`Others`

Default	No default value
Property type	String
Usage	Some SQL statements require special processing, which varies by database type. For example, the Struct and Array data types in Oracle require special processing. This property specifies the RDBMS that is used, which determines the database type. For other databases, 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 102. Delivery type details

Required	No
Possible values	`ORDERED` `UNORDERED`

Table 102. Delivery type details (continued)

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. If you set to `UNORDERED`, you must specify multiple connections for the event delivery to allow the adapter to deliver events to the end-point correctly. To specify multiple connections, edit the export file and specify appropriate values for the "Minimum connections (MinimumConnections)" on page 268 and "Maximum connections (MaximumConnections)" on page 267 properties.
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 103. Do not process events that have a timestamp in the future details

Required	Yes
Possible values	`True` `False`
Default	`False`
Property type	Boolean
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 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 104. Ensure once-only event delivery details

Required	Yes
Possible values	`True` `False`
Default	`True`
Property type	Boolean

Table 104. Ensure once-only event delivery details (continued)

Usage	When this property is set to `True`, the adapter provides assured once event delivery. This means that each event will be delivered once and only once. A value of `False` does not provide assured once event delivery, but provides better performance. When this property is set to `True`, the adapter attempts to store transaction (XID) information in the event store. If it is set to `False`, the adapter does not attempt to store the information. This property is used only if the export component is transactional. If 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 105. 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
Bidi supported	Yes

Event query type for processing events (EventQueryType)

This property specifies whether to use standard or custom query processing.

Table 106. Event query type details

Required	Yes
Possible values	`Standard` `User-Defined(Dynamic)`
Default	`Standard`
Property type	String

Table 106. Event query type details (continued)

Usage	The valid values are `Standard`, for standard event processing, and `Dynamic`, for custom event processing. If this property is set to `User-Defined(Dynamic)`, the CustomEventQuery, CustomUpdateQuery, and CustomDeleteQuery properties are used. If this property is set to `Standard`, these 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 107. 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 is 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 108. Event types to process details

Required	No
Possible values	A comma-delimited (,) list of business object types
Default	`null`
Property type	String
Usage	Events are filtered by business object type. If the property is set, the adapter delivers only those events that are in the list. A value of `null` indicates that no filter will be applied and that all events will be delivered to the export.

Table 108. Event types to process details (continued)

Example	To receive events related to the Customer and Order business objects only, specify this value: `Customer,Order` This property applies only to standard event processing. For custom event processing, the custom event query must perform necessary filtering. If the EventTypeFilter and AdapterInstanceEventFilter properties are both set, the adapter processes only events that meet both criteria. That is, it processes only those events whose type is specified in the EventTypeFilter property and whose connector_ID column matches the AdapterInstanceEventFilter property.
Globalized	No
Bidi supported	No

Retry limit for failed events (FailedEventRetryLimit)

This property specifies the number of times that the adapter attempts to redeliver an event before marking the event as failed.

Table 109. Retry limit for failed events details

Required	No
Possible values	Integers
Default	5
Property type	Integer
Usage	Use this property to control how many times the adapter tries to send an event before marking it as failed. It accepts the following values: **Default** If this property is not set, the adapter tries five additional times before marking the event as failed. **0** The adapter tries to deliver the event an infinite number of times. When the property is set to `0`, the event remains in the event store and the event is never marked as failed. **> 0** For integers greater than zero, the adapter retries the specified number of times before marking the event as failed. **< 0** For negative integers, the adapter does not retry failed events.
Globalized	No
Bidi supported	No

JDBC driver class (JDBCDriverClass)

This property specifies the class name of the JDBC driver that is used to connect to the database.

Table 110. JDBC driver class details

Row	Explanation
Required	Yes, when the connection is of the type LocalConnectionProps or ConnectionProps.
Possible values	Values are database-specific.
Default	No default value
Property type	String
Usage	In the external service wizard, the JDBC driver class is displayed 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 type the database 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.

Examples	Values for the JDBC driver class is displayed in both the external service wizard and the administrative console. The following examples depict the JDBC driver class properties in both the external service wizard and the administrative console.**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

Maximum connections (MaximumConnections)

This property specifies the maximum number of connections that the adapter can use for inbound event delivery.

Table 111. Maximum connections details

Required	No

Table 111. Maximum connections details (continued)

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 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 112. 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 113. Password details

Required	No. For inbound processing, if you set the Authentication alias or the DataSourceJNDIName, the password is not mandatory. However, if you set the DataSourceJNDIName, and the Password field, the value specified for the Password takes precedence. For outbound processing, if you set the Authentication alias, XADataSourceJNDIName or PoolDataSourceJNDIName property, the password is not mandatory. However, if you set the XADataSourceJNDIName or PoolDataSourceJNDIName property, and the Password field, the value specified for the Password takes precedence.
Default	No default value
Property type	String

Table 113. Password details (continued)

Usage	For inbound processing, setting this property overrides the password specified for the data source on the server using the Authentication alias or DataSourceJNDIName property. For outbound processing, setting this property overrides the password specified for the data source on the server using the Authentication alias, or XADataSourceJNDIName or PoolDataSourceJNDIName property. If you specify JAAS as the security credential, the authentication alias will override this property.
Globalized	Yes
Bidi supported	Yes

SQL query to verify the connection (PingQuery)

This property specifies the SQL query that is used to test the reliability of the connection to the database.

Table 114. 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. For inbound event processing, when the ping query indicates an invalid connection, the adapter throws an exception.
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 115. 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 116. 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, fewer events are processed per polling period and the adapter's performance might improve slightly.
Globalized	No
Bidi supported	No

Query timeout (seconds) (QueryTimeOut)

This property specifies the maximum number of seconds a query can take to run all SQL statements.

Table 117. 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	No
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 118. Retry interval details

Required	Yes
Default	2000
Unit of measure	Milliseconds
Property type	Integer

Table 118. Retry interval details (continued)

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	No
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 119. Number of times to retry the system connection details

Required	No
Possible values	`0` and positive integers
Default	`0`
Property type	Integer
Usage	This property controls how many times the adapter retries the connection if the adapter cannot connect to the database to perform inbound processing. A value of `0` indicates an infinite number of retries. To control whether the adapter retries if it cannot connect to the database when it is first started, use the RetryConnectionOnStartup property.
Globalized	No
Bidi supported	No

Retry EIS connection on startup (RetryConnectionOnStartup)

This property controls whether the adapter attempts to connect again to the database if it cannot connect at startup.

Table 120. Retry EIS connection on startup details

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

Table 120. Retry EIS connection on startup details (continued)

Usage	This property indicates whether the adapter should retry the connection to the database if the connection cannot be made when the adapter is started: • Set the property to `False` when you want immediate feedback about whether the adapter can establish a connection to the database, for example, when you are building and testing the application that receives events from the adapter. If the adapter cannot connect, the adapter writes log and trace information and stops. The administrative console shows the application status as `Stopped`. After you resolve the connection problem, start the adapter manually. • Set the property to `True` if you do not need immediate feedback about the connection. If the adapter cannot connect during startup, it writes log and trace information, and then attempts to reconnect, using the RetryInterval property to determine how frequently to retry and the value of the RetryLimit property to retry multiple times until that value is reached. The administrative console shows the application status as `Started`.
Globalized	No
Bidi supported	No

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 121. 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 the ReturnDummyBOForSP property is set to `True`, a dummy business object is created and the returned parameters (including the out and inout parameters) from the stored procedure are populated into the corresponding attributes
Globalized	No
Bidi supported	No

Stop the adapter when an error is encountered while polling (StopPollingOnError)

This property specifies whether the adapter will stop polling for events when it encounters an error during polling.

Table 122. 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 123. Stored procedure to run after poll details

Required	No
Default	No default value
Property type	String

Usage	The stored procedure takes one input parameter of type integer for poll quantity, which is set by the adapter at run time. Here is the syntax:
	`call <spname> (?)`
	
	The stored procedure can also have additional input parameters, and these values must be specified in the call statement itself. Here's the syntax for a stored procedure that has additional input parameters:
	`call <spname> (25, ?, 'test')`

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 124. Stored procedure to run before poll details

Required	No
Default	No default value

Table 124. Stored procedure to run before poll details (continued)

Property type	String

Usage	The stored procedure takes one input parameter of type integer for poll quantity, which is set by the adapter at run time. Here is the syntax:
	`call <spname> (?)`
	The stored procedure can also have additional input parameters, and these values must be specified in the call statement itself. Here's the syntax for a stored procedure that has additional input parameters:
	`call <spname> (25, ?, 'test')`

Globalized	Yes
Bidi supported	Yes

User name (UserName)

This property specifies the database user name that is used to access the database.

Table 125. User name details

Required	No. For inbound processing, if you set the Authentication alias or the DataSourceJNDIName, the User name property is not mandatory. However, if you set the DataSourceJNDIName, and User name field, the value specified for the User name takes precedence. For outbound processing, if you set the Authentication alias, XADataSourceJNDIName or PoolDataSourceJNDIName property, the User name is not mandatory. However, if you set the XADataSourceJNDIName or PoolDataSourceJNDIName property, and the User name field, the value specified for the User name takes precedence.
Default	No default value
Property type	String
Usage	For inbound processing setting this property overrides the user name specified for the data source on the server using the DataSourceJNDIName property or the Authentication alias. For outbound processing, setting this property overrides the user name specified for the data source on the server using the XADataSourceJNDIName property, PoolDataSourceJNDIName property or the Authentication alias. If you specify JAAS as the security credential, the authentication alias will override this property.
Globalized	Yes
Bidi supported	Yes

Trim spaces for key-value pair (performTrimOnObjectKeyValue)

This property specifies if the inbound operation has to trim the spaces before and after a key-value pair.

Table 126. Trim spaces for key-value pair details

Required	No.
Possible values	`True` `False`
Default	`True`
Property type	Boolean
Usage	If you set this property to `True`, the inbound operation trims the spaces before and after a key-value pair. If you set this property to `False`, the spaces are not trimmed.
Globalized	No
Bidi supported	No

Separate key-value pair (nameValuePairDelimiter)

This property specifies the delimiter to separate the key-value pair.

Table 127. Separate name-value pair details

Required	No.
Possible values	Any string
Default	N/A
Property type	String
Usage	Delimiter to separate key-value pair.
Example	key1=value1;key2=value2;key3=value3 or value1;value2;value3
Globalized	No
Bidi supported	No

Separate key and value (valueDelimeter)

This property specifies the delimiter to separate the key and value in a key-value pair.

Table 128. Separate key and value details

Required	No.
Possible values	Any string
Default	=
Property type	String
Usage	Delimiter to separate the key and value.
Example	key1=value1 or value1
Globalized	No
Bidi supported	No

Unset value (unsetValueKeyWord)

This property represents the unset value in the object_key field.

Table 129. Unset value key word details

Required	No
Possible values	JCA_JDBC_UNSET
Default	blank
Property type	String
Usage	If this property is set as not blank, the adapter checks for the value in the object_key field. If a value exists in the object_key field, the adapter treats this field as unset. For example, if you set this property to “JCA_JDBC_UNSET” and the object_key field value is "1; JCA_JDBC_UNSET;2", during run time, the adapter treats the field corresponding to the value "JCA_JDBC_UNSET" as unset and it does not use that field in the WHERE clause for retrieving the object.
Globalized	No
Bidi supported	No

Null value (nullValueKeyWord)

This property represents the null value in the object_key field.

Table 130. Null value key word details

Required	No
Possible values	JCA_JDBC_NULL
Default	blank
Property type	String
Usage	If this property is set as not blank, the adapter checks for the value in the object_key field. If a value exists in the object_key field , the adapter treats this field as null. For example, if you set this property to “JCA_JDBC_NULL” and the object_key field value is "1; JCA_JDBC_NULL;2", during run time, the adapter treats the field corresponding to the value "JCA_JDBC_NULL" as null and it sets that field value to ‘is Null' in the WHERE clause when retrieving the object.
Globalized	No
Bidi supported	No

Database connection information (ConnectionType)

This property specifies how the adapter establishes connection to the database.

Table 131. Database connection information

Required	Yes
Possible values	ConnectionProps or DataSourceJNDI
Default	ConnectionProps
Property type	String

Table 131. Database connection information (continued)

Usage	This property specifies how adapter establish database connection at run time. This property can have the following values: • ConnectionProps - Indicates that the database connection is established using the DatabaseURL and JDBCDriverClass properties. • DataSourceJNDI - Indicates that the database connection is established using the DataSourceJNDIName property corresponding to predefined datasource. For a new application, this property is automatically set by external service wizard. For a migrated application, this property is set during the migration process according to the ActivationSpecification properties. • If the DataSourceJNDIName property is set, the value of this property is set to DataSourceJNDI. • If the DataSourceName is set, the value of this property is set to ConnectionProps. If this property is not set, adapter uses backward compatibility mode to establish database connection. In backward compatibility mode, 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. 2. If the DataSourceJNDIName is not set, then the adapter uses the DatabaseURL, JDBCDriverClass, UserName and Password properties to establish the connection.
Globalized	No
Bidi supported	No

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 multibyte 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 right-to-left (Hebrew or Arabic, for example) and left-to-right (a URL or file path, for example) semantic content within the same file.

Globalization

Globalized software applications are designed and developed for use within multiple linguistic and cultural environments rather than a single environment. WebSphere Adapters, WebSphere Integration Developer, WebSphere Process Server, and WebSphere Enterprise Service Bus are written in Java. The Java run time 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 is 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.

The bidirectional format consists of five attributes. When you set bidirectional properties, you assign values for each of these attributes. The attributes and settings are listed in the following table.

Table 132. Bidirectional format attributes

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

Table 132. Bidirectional format attributes (continued)

Letter position	Purpose	Values	Description	Default setting
5	Numeric Shaping	H	National (Hindi)	N
		C	Contextual shaping	
		N	Numbers are not shaped (Nominal)	

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 the types of bidirectional properties.

Table 133. Bidirectional property types

Property type	Data transformations
EIS	Controls the format for content data, or data that is sent by the enterprise information system, which 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, which 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 the external 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

Fault business objects

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, and are caused by a business rule violation or a constraint violation.

The WebSphere Adapter for JDBC enables faults for you. Manual configuration of faults is not required. The WBIFault business object contains information that is required to handle the fault.

The adapter provides the following fault business objects that the wizard creates:

- IntegrityConstraintFault

  When processing a Create operation, the adapter generates this fault if the database generates the SQLException exception for an integrity constraint violation. For example, if a foreign key is not found, the adapter generates this fault.
- MatchesExceededLimitFault

  When processing a RetrieveAll operation, the adapter generates this fault if the number of records returned from the database query exceeds the value for the MaxRecords property in the interaction specification.

  To increase the number of records that are returned, increase the value of the MaxRecords property in the interaction specification properties for the RetrieveAll operation.

  The business object for this fault has only 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, the adapter generates this fault. This fault can occur for the Create, Delete, Update, Retrieve, ApplyChanges and Exists operations.
- MultipleMatchingRecordsFault

  When processing a Retrieve or Update operation, the adapter generates 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 generates this fault.
- RecordNotFoundFault

  When retrieving data, the adapter generates 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 or Update operation, the adapter generates this fault if it receives a SQLException exception from the database because of a unique constraint violation.

The following notes apply to the business faults that are associated with this adapter:

- The IntegrityConstraintFault is defined for Create and Execute operations only.

- The RetrieveAll operation for MatchesExceededLimitFault is for both table business objects and query business objects.
- The MissingDataFault is defined only for Create, Delete, Update, Retrieve, Exists, and ApplyChanges operations.
- The ObjectNotFoundFault is defined for Create, Update, and Delete operations only.
- For the RecordNotFoundFault the RetrieveAll operation is used for both table business objects and query business objects.
- The UniqueConstraintFault is defined only for Create and Execute operations.

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/v7r0mx/topic/com.ibm.wbit.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 CWYDB are issued by the common component shared by WebSphere Adapter for JDBC and WebSphere Adapter for Oracle E-Business Suite
- 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

To help you use WebSphere Adapters, samples and tutorials are available from the Business Process Management Samples and Tutorials Web site. You can access the samples and tutorials in either of the following ways:

- 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 6.2.x, information center, which includes WebSphere Process Server, WebSphere Enterprise Service Bus, and WebSphere Integration Developer information: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp
- WebSphere Adapters, version 6.1.x, information center: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp
- 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, 2009

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department 2Z4A/SOM1
294 Route 100
Somers, NY 10589-0100
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: (c) (your company name) (year). Portions of

this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning:

Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A complete and current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

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

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

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

Other company, product, or service names may be trademarks or service marks of others.

This product includes software developed by the Eclipse Project (http://www.eclipse.org).

Index

A

activation specification properties
  list of 254
  setting in administrative console 159, 163
adapter 69
  project, create 72
adapter application
  starting 165
  stopping 165
Adapter for JDBC
  administering 155
Adapter for JDBC module
  exporting as EAR file 152
  installing EAR file on server 153
  starting 165
  stopping 165
adapter implementation
  security 49
adapter messages 283
adapter performance 166
adapter sharing 192
adapter technotes 284
after-image 5
application-specific information 207
  adding to object 80, 114
  for attributes of type child business object 204
  for simple attributes 199
ApplyChanges operation 12
archive events 194
artifact 69
assembly editor, modifying adapter application-specific information 110, 134
assured once delivery 18
attribute properties 197
attribute type, business object 198
authentication 69
  description 50
  external service wizard 50
  run time 51
authentication alias 51, 69, 71

B

batch SQL business object 46
  structure 29
BIDI support
  schema name in Oracle 182
  stored procedure in MS SQL 181
BPEL 191
business faults 48, 191, 282
business faults:business objects 282
business graph 4
business integration adapters to JCA-compliant adapters 60
business object 69
business object information 197
business object structure
  for batch SQL business objects 29
  for query business objects 28
  for stored procedure business objects 25
  for table or view business objects 23
business object structure *(continued)*
  for wrapper business objects 30
business objects 22, 207
  attribute types 198
  attributes 197
  batch SQL 46
  cardinality 32
  composite keys 110, 134
  how to view 81, 115
  multiple parents 110, 134
  naming conventions 210
  query 46
  stored procedure 38

C

cardinality 32, 197
CCSID 184
CEI (Common Event Infrastructure) 169
changing schema name 180
CL 69
clustered environment
  deploying in 53
  description 53
  inbound processes 54
  outbound processes 54
Common Event Infrastructure (CEI) 169
compatibility matrix 2
complex data types 39
composite key 193
  object_key 193
confidential data, disguising 49
confidential tracing 49
configuration overview 69
configuring
  logging 171
  Performance Monitoring Infrastructure (PMI) 166
  tracing 171
configuring JDBC adapter for XA 105, 106
connector project 72
control language 69
Create operation 6
custom properties
  activation specification 159, 163
  managed connection factory 157, 162
  resource adapter 155, 161
custom queries
  standard SQL 19
  stored function 19
  stored procedure 19

D

data types
  complex 39
database sequence
  DB2 110, 134
  Microsoft SQL Server 110, 134
  Oracle 110, 135
Date 191

© Copyright IBM Corp. 2006, 2009

DB2
  sample scripts 70
debugging
  self-help resources 196
Delete operation 14
delta 5
deployment
  environments 143
  options 51
  to production environment 148
  to test environment 143
deployment environment 69
developerWorks 284
developerWorks resources, WebSphere Adapters 283
distributed transactions, see *XA* transactions 235

E

EAR file
  exporting 152
  installing on server 153
EBCDIC 184
edit binding
  import component 137
education, WebSphere Adapters 283
embedded adapter
  activation specification properties, setting 159
  considerations for using 53
  description 51
  managed connection factory properties, setting 157
  resource adapter properties, setting 155
embedded deployment 145
enableHASupport property 54
event delivery 263
event processing
  custom 4, 18
  standard 4, 17
event query
  user-defined 194
event store 20, 21
event store set up 70
Execute operation 14
Exist operation
  limitations 15
Exists operation 15
  for database table business objects 15
  for database views business objects 15
  for nicknames business objects 15
  for synonyms business objects 15
export components 179
export file 69
exporting module as EAR file 152
external dependencies, adding 143, 145
external service connection properties 213, 242
external service discovery
  attribute information 197
external service wizard
  authentication in 50
  connection properties 76
  start up 76

F

fault business object 282
faults
  description 48, 282
FFDC (first-failure data capture) 174
files
  SystemOut.log log file 173
  trace.log trace file 173
first-failure data capture (FFDC) 174
flat business objects 32
foreign key 197

G

global transaction 105, 106, 183
  DB2 6, 107
  Oracle 6, 107
  with XA DataSource JNDI name 106
  with XADataSourceName and XADatabaseName 107

H

hardware and software requirements 2
hardware requirements 2
hierarchical business objects 32
high-availability environment
  deploying in 53
  description 53
  inbound processes 54
  outbound processes 54

I

i5/OS 184
  sample scripts 183
IBM WebSphere Adapter Toolkit 284
identity column
  DB2 110, 134
  Informix 110, 135
  Oracle 110, 134
identity columns
  description 23
implementation, Java 146
import component 137
import file 69
inbound configuration properties 240
inbound processing 3, 69
  native method 195
installing EAR file 153
interaction specification properties
  changing 141
interaction specification property 239
interface 69
iterative development
  connection-based editing 137
  Edit Binding 137
  import component 137

J

J2C connection factory
  see managed connection factory 225
JAR file, adding external 143, 145
Java 2 security 49, 51
Java implementation 146
JDBC 2.0 1
JDBC driver
  DB2 74
  MySQL 183

JDBC driver *(continued)*
Oracle 74
SQL Server 74
JDBC driver files 73
JRE 73
JRE version
JDBC driver 1
runtime environment 1

L

local connection
pool datasource 107
with Database URL 107
local transactions 5
locks 183
Log Analyzer 171
Log and Trace Analyzer, support for 47
log and trace files 47
log files
changing file name 173
disabling 171
enabling 171
level of detail 171
location 173
logging
configuring properties with administrative console 171

M

managed (J2C) connection factory properties
setting in administrative console 157, 162
managed connection factory properties
details 225
XA data source name 234
XADataSourceName 234
matrix, compatibility 2
MaxNumOfRetRS 190
messages, adapter 283
Metadata selection properties
how to specify (inbound) 124
how to specify (outbound) 100
method bindings 195
migration 60
WebSphere InterChange Server migration wizard 63
migration considerations 55
migration overview
WebSphere InterChange Server applications 61
module 69
monitoring performance 166
MSSQL
sample scripts 70
multiple connection 263
multiple export components 179
multiple stored procedures 191
MySQL 183

N

naming conventions for business objects 210
native method 195
NULL objects
retrieving 11

O

object 69
object_key
event table 193
timestamp 201
operation 69
operations
ApplyChanges 12
Create 6
Delete 14
Execute 14
Exists 15
Retrieve 7
RetrieveAll 8
Update 11
Oracle
sample scripts 70
outbound configuration properties 212
outbound operations
list of 5
outbound processing 2, 69

P

package files for adapters 172
performance
prepared statement caching 55
Performance Monitoring Infrastructure (PMI)
configuring 166
description 166
viewing performance statistics 168
performance statistics 168
PMI (Performance Monitoring Infrastructure)
configuring 166
description 166
viewing performance statistics 168
polling 20
PoolDataSourceJNDIName 6
prepared statement caching 55, 237, 239
primary key 197
problem determination
self-help resources 196
solutions to common problems 174
project 69
project interchange (PI) file
project interchange files 59
projects 59
updating without migrating 59
properties
activation specification 159, 163
list of 254
configuration properties
inbound 240
outbound 212
external service connection 213, 242
inbound configuration 240
managed (J2C) connection factory 157, 162
outbound configuration 212
resource adapter 155, 161

Q

query 69
query business object
generate from SELECT statement 46
structure 28

R

RAR (resource adapter archive) file
  description 149
  installing on server 149
RecordNotFoundException 7, 9, 189, 190, 230
Redbooks, WebSphere Adapters 283
related information 283
related products, information 283
requirements, hardware and software 2
resource adapter archive (RAR) file
  description 149
  installing on server 149
resource adapter properties
  details 220, 249
  setting in administrative console 155, 161
ResourceException
  inbound processing 188
  outbound processing 188
Retrieve operation 7
RetrieveAll operation
  for database table business objects 8
  for user-specified query business objects 10
Retry limit property 271
road map for configuring the module 69
roadmap for migrating
  WebSphere InterChange Server applications 60
runtime environment
  authentication in 51
  deploying EAR file to 148

S

sample scripts 70
samples 67
SCA module 179
schema name
  modify 180
security
  disguising sensitive data 49
security features
  adapter 49
  Java 2 security 49
security, Java 2 51
self-help resources 196
sensitive data, disguising 49
serial columns
  description 23
service 69
service interface queue 70
software dependencies 73
software dependencies, adding external 143, 145
software requirements 2
SQL keywords
  column name 182
  table name 182
SQL Server 2000 JDBC driver 179
stand-alone adapter
  activation specification properties, setting 163
  considerations for using 53
  description 51
  managed connection factory properties, setting 162
  resource adapter properties, setting 161
starting adapter applications 165
stopping adapter applications 165
stored functions
  overview 45
stored procedure 14
  business object structure 25
  definition 38
  iSeries DB2 180
  return value 184
  sample of definition 44
  unique name 38, 89
  with same names 38, 89
stored procedure business object 38
stored procedures
  overview 38
  SQL statements 38
structure of business objects 23
support
  overview 171
  self-help resources 196
  technical 284
SystemOut.log file 173

T

table
  business object structure 23
target component 145
technical support 284
technotes 2, 196, 284
technotes, WebSphere Adapters 283
test environment
  adding module to 147
  deploying to 143, 147
  testing modules 147
Time 191
TimeStamp 191
trace files
  changing file name 173
  disabling 171
  enabling 171
  level of detail 171
  location 173
trace.log file 173
tracing
  configuring properties with administrative console 171
transaction timeout 183
transactions 5
  with XADataSourceJNDIName 6
transactions, see also XA transactions and local transactions 5
triggers on user tables 71
troubleshooting
  overview 171
  self-help resources 196
tutorials 67

U

UDF, see user-defined function 45
UID 110, 134
UNORDERED 263
Update operation 11
update schema name 180
user-defined function 45

V

view
  business object structure 23

W

WebSphere Adapters, version 6.0, information 284
WebSphere Adapters, version 6.1.x, information 284
WebSphere Application Server information 284
WebSphere business integration adapters 60
WebSphere Business Integration Adapters information 284
WebSphere Business Process Management, version 6.2.x, information 284
WebSphere Enterprise Service Bus
  information 284
WebSphere Extended Deployment 54
WebSphere Integration Developer
  information 284
  test environment 143
WebSphere Process Server
  information 284
WebSphere Process Server or WebSphere Enterprise Service Bus
  deploying to 148
wiring components 145
wrapper business object
  structure 30
wrapper business objects
  creating 101, 124

X

XA transaction 105
  DB2 6, 107
  Oracle 6, 107
  with XA DataSource JNDI name 106
  with XADataSourceName and XADatabaseName 107
XA transactions 5
  DB2 databases 6
  Oracle databases 6
  XA data source name 235
XML data types 191
XQueries 191

Z

z/OS
  sample scripts 183
zOS
  sample scripts 70

Printed in USA